

Please quote as: Koch, M.; Hoffmann, A. & Hoffmann, H. (2014): Usability-Anforderungsmuster für die Entwicklung von Software-Applikationen. In: Working Paper Series, Nr. 6, Kassel, Germany.





Working Paper Series

Kassel University
Chair for Information Systems
Prof. Dr. Jan Marco Leimeister

Nr. 6

Matthias Koch,
Axel Hoffmann, Holger Hoffmann

Usability-Anforderungsmuster für die Entwicklung von Software-Applikationen

Kassel, Januar 2014

Series Editor:

Prof. Dr. Jan Marco Leimeister, Kassel University
Chair for Information Systems

Pfannkuchstr. 1, 34121 Kassel, Germany
Tel. +49 561 804-6068, Fax: +49 561 804-6067

leimeister@uni-kassel.de

http://www.wi-kassel.de

Zusammenfassung

Ziel der vorliegenden Arbeit ist es, einen Katalog mit Anforderungsmustern für das frühzeitige Berücksichtigen von Usability in der Softwareentwicklung zu erstellen. Dazu wird eine Literaturrecherche durchgeführt, in der die gängigsten Usability-Gestaltungsbereiche betrachtet werden. Auf dieser Grundlage werden 47 Usability-Anforderungsmuster im Sinne des Requirements Engineerings aufgestellt. Die erstellten Muster behandeln eine breite Auswahl an Themen wie beispielswiese Nutzerkontrolle, Effizienz und Fehler-Handhabung. Um die praktische Anwendbarkeit und die Qualität der Muster zu prüfen, werden Gespräche mit Software-Entwicklern geführt. In diesen Gesprächen wird bestätigt, dass Usability für Praktiker eine sehr hohe Bedeutung hat. Allerdings wird bei den befragten Unternehmen Usability bereits in begleitender Form während der gesamten Entwicklung berücksichtigt, ohne dass eigene Anforderungen dafür aufgestellt werden. Die Befragten begrüßen den Katalog mit Anforderungsmustern als vollständigen und hilfreichen Leitfaden, um bestehende Anforderungskataloge und Designs zu überprüfen. Die Muster werden außerdem an einem Fallbeispiel angewendet, um Usability-Anforderungen für eine Smartphone-App aufzustellen und Gestaltungsvorschläge zu ermitteln. Durch das Prüfen sämtlicher Muster können Schwächen des Designs aufgedeckt und Lösungsvorschläge erarbeitet werden.

Stichworte: Usability, Benutzbarkeit, Anforderungsmuster, Software-Anforderungen, Requirements Engineering, Interface Design.

1 Einleitung

Mit dem Aufstieg von internetfähigen Mobilgeräten wie Smartphones und Tablet-PCs wurde der Markt für Software-Anwendungen neu belebt: Applikationen für die neuen Gerätetypen sind ebenso populär wie mobiloptimierte Designs im klassischen Internetbereich (vgl. Alby 2008, 103ff.). Im Kampf um die Aufmerksamkeit und Treue des Endnutzers spielt die Usability von Software eine große Rolle. Usability ist die Benutzbarkeit eines Systems. In ihr drückt sich aus, wie effektiv, effizient und zufriedenstellend Endnutzer eine Software in einer bestimmten Situation verwenden können (vgl. International Standards Organization 2006). Die Erwartungen der Endnutzer sind diesbezüglich stark gestiegen: neben der Verbreitung klassischer Computer hat der Erfolg von intuitiven, benutzerfreundlichen mobilen Anwendungen die Messlatte für neue Softwareprodukte erhöht. Oft entscheidet die Benutzbarkeit einer Software darüber, ob diese am Markt besteht oder nicht (Kaasinen 2005, 1f.). Umso wichtiger ist es für Entwickler und Gestalter, Usability so früh wie möglich in die Entwicklung neuer Software miteinzubeziehen. Viele der Faktoren, die zu guter Benutzbarkeit beitragen, werden bereits zu Beginn einer Softwareentwicklung bearbeitet und sind im Nachhinein schwer änderbar (vgl. Holzinger 2005, 72). Es entstanden deshalb mehrere Arbeits- und Forschungsfelder für diesen Bereich. Die Methoden der Usability-Evaluation und des Usability Engineerings legen den Entwicklungsfokus von Anfang an auf Benutzbarkeit und beziehen den Endnutzer beispielsweise in Form von Software-Tests in die Entwicklung mit ein (vgl. Nielsen 1994, 1ff.; Rosson/Carrol 2002, 14f.). Viele dieser Methoden gelten in der Praxis – obgleich ihres großen Nutzens – als zeitaufwendig und teuer. Es wird daher nach einfacheren Wegen gesucht, Usability systematisch zu bearbeiten (vgl. Blair-Early/Zender 2008, 85ff.). Dies kann durch die Arbeitsmethoden des sog. „Requirements Engineerings“ geschehen, dem Entwickeln nach Anforderungen. Mit Hilfe von Anforderungen können eindeutige Ziele an ein zu schaffendes Produkt gestellt werden. Da gute Usability ein immer wiederkehrendes Softwareziel ist, empfiehlt sich das Verwenden von speziellen Vorlagen, sog. „Anforderungsmustern“. Mit Anforderungsmustern können Anforderungen für immer wieder auftachende Problembereiche schnell und präzise formuliert werden (Withall 2007, 19f.). Sie stellen nicht nur eine Formulierhilfe, sondern auch eine Wissensbasis dar, mit der Entwickler von Anfang an bestimmte Entwicklungsaspekte berücksichtigen und

durchdenken können. Bisher gibt es allerdings nur wenige Versuche, Usability in Form von Anforderungen oder Anforderungsmustern festzuhalten.

Ziel der vorliegenden Arbeit ist es, einen Katalog mit Anforderungsmustern für Usability zu erstellen. Mit Hilfe dieser Anforderungsmuster soll Usability so früh und so umfassend wie möglich in Softwareentwicklungsprozessen berücksichtigt werden können. Die erstellten Anforderungsmuster sollen sowohl theoretisch fundiert als auch praktisch anwendbar sein. Sie werden daher auf Grundlage einer Literaturrecherche erarbeitet und durch Expertengespräche und die Anwendung an einem Fallbeispiel geprüft.

Die vorliegende Arbeit verfolgt im Wesentlichen drei Forschungsfragen, die auch die Gliederung des Hauptteils vorgeben:

1. Welche Software-Anforderungen für Usability gibt es und welche Anforderungsmuster lassen sich daraus ableiten?
2. Wie gut sind diese Anforderungsmuster in der Praxis anwendbar?
3. Inwieweit lassen sich die Anforderungsmuster an einem konkreten Fallbeispiel anwenden, um Gestaltungsvorschläge zu erarbeiten?

Die Beantwortung der Forschungsfragen beginnt mit einer Beschreibung der begrifflichen Grundlagen in Kapitel 2. Darin werden zuerst Anforderungsmuster sowie die Arbeitsdisziplinen rund um die Arbeit mit Anforderungen beschrieben. Danach folgt eine genaue Klärung des Begriffs Usability. In Kapitel 3 wird das forschungsmethodische Vorgehen beschrieben, welches sich in die Vorgehensweisen der Literaturrecherche, der Expertenbefragung und der praktischen Anwendung der Muster an einem Prototyp aufteilt.

Um die Usability-Anforderungsmuster zu erstellen, wird in Kapitel 4 eine Literaturrecherche in den Bereichen Softwareentwicklung, Produktdesign, Interfacedesign und Software-Ergonomie durchgeführt. Mit Hilfe dieser Recherche werden Gestaltungsempfehlungen zusammengetragen, aus denen Anforderungen und Anforderungsmuster abgeleitet werden. In den Kapitel 4.1 bis 4.15 werden 15 unterschiedliche Gestaltungsbereiche diskutiert und dazugehörige Anforderungsmuster aufgestellt. In Kapitel 4.16 werden die Muster im Überblick betrachtet und ihr Zusammenspiel untersucht. Um die praktische Anwendbarkeit der Muster zu prüfen, wurden Expertengespräche mit Praktikern geführt. Experten aus drei Unternehmen,

die selbst Requirements Engineering anwenden, bewerteten die Usability-Anforderungsmuster nach ausgewählten Qualitätskriterien. Kapitel 5 enthält die zusammengefassten Gesprächsergebnisse. Als weitere Qualitätsüberprüfung der Muster werden diese in Kapitel 6 an dem Prototyp einer Smartphone-Applikation – einem Restaurant-Finder – angewendet. Dazu werden zunächst passende Usability-Anforderungen für die Applikation zusammengestellt und auf deren Grundlage Gestaltungsvorschläge erarbeitet, die diese Anforderungen berücksichtigen. Kapitel 7 enthält eine abschließende Zusammenfassung der Ergebnisse und einen Ausblick. Die vollständigen Usability-Anforderungsmuster befinden sich in Anhang A.

2 Grundlagen zu Anforderungsmustern und Usability

Im folgenden Kapitel werden die theoretischen Grundlagen der Arbeit erklärt. Dabei handelt es sich zum einen um die Arbeit mit Anforderungsmustern, zum anderen um den Bereich der Usability.

2.1 Anforderungsmuster

Anforderungsmuster sind eine Arbeitsunterstützung in der Softwareentwicklung bei der Definition von Anforderungen (vgl. Renault et al. 2009, 2). Eine Anforderung ist eine einzelne, überprüfbare Funktion oder Eigenschaft, die ein System haben muss (Withall 2007, 4). Anforderungen beschreiben, „was der Kunde oder Benutzer vom Produkt erwartet" (Ebert 2012, 11). Sie stellen damit konkrete Vorgaben, Eigenschaften, Bedingungen und Ziele dar, die Anbieter beim Erstellen ihrer Leistung erreichen müssen (Pohl 2008, 13f.). Die vorliegende Arbeit bezieht sich konkret auf das Entwickeln von Software-Applikationen, weshalb hier speziell die Anforderungen von Softwareprodukten und IT-Dienstleistungen betrachtet werden. Software-Applikationen in diesem Sinne sind Programme oder Programmkomponenten für Computer, Tabletts, Smartphones oder jedes andere technische Gerät, das Programme abspielen kann (vgl. Alby 2008, 103ff.; Ebert 2012, 318f.). Eine zusätzliche Voraussetzung ist, dass die Software eine Benutzeroberfläche hat. Programme, mit denen Endnutzer nicht interagieren (wie z. B. Treiber oder Plug-ins), werden nicht betrachtet.

Erarbeitet eine Gruppe Entwickler wiederholt Projekte unter Nutzung von Anforderungen, ist es wahrscheinlich, dass sich bestimmte Anforderungen bei jedem Projekt wiederholen. Oft wird daher beim Erstellen neuer Anforderungskataloge auf alte Kataloge zurückgegriffen, indem alte Anforderungen kopiert oder abgeändert werden. Das Erstellen immer neuer Anforderungskataloge bleibt hierbei jedoch mühselig, da die Anforderungen noch nicht standardisiert sind (vgl. Renault et al. 2009, 2). Um das Erstellen von sich wiederholenden oder sich ähnelnden Anforderungen zu erleichtern, empfiehlt sich das Verwenden spezieller Vorlagen in Form von Anforderungsmustern (vgl. Franch et al. 2010, 85f.).

Ein Anforderungsmuster ist ein Ansatz zur Spezifizierung eines bestimmten Anforderungstyps (Withall 2007, 19). Mit Hilfe eines Anforderungsmusters kann eine Anforderung eines bestimmten Typs für ein spezielles System abgeleitet und

ausformuliert werden. Das Verwenden von Anforderungsmustern hat gegenüber dem herkömmlichen Aufstellen von Anforderungen zwei Vorteile: Zum einen sind die Muster Leitfäden, die verschiedene Empfehlungen enthalten und den Anwendern damit eine wertvolle Hilfestellung bieten. Zum anderen spart ihre Verwendung Zeit, da sie eine Grundstruktur zum Formulieren der Anforderungen bieten. Hinzu kommt, dass Anforderungen eines Typs konsistenter sind, wenn sie auf Grundlage eines Musters erstellt wurden (vgl. Withall 2007, 19).

Auch bei der Verwendung von Mustern kann zwischen der Ebene der Anforderungen und der Lösungsebene unterschieden werden: Lösungsmuster, auch Design-Patterns genannt, sind in der Softwareentwicklung ein gebräuchliches Instrument zur Formulierung wiederkehrender Lösungsschemata (Gamma et al. 1995, 2).

Ziel der vorliegenden Arbeit ist es, Anforderungsmuster für Usability zu erstellen. Um die Disziplin der Anforderungserstellung anzuwenden, müssen zunächst der Aufbau und die Elemente eines Anforderungsmusters erklärt werden.

Dazu wird mit den Anforderungen selbst begonnen. Eine vollständige Anforderung besteht aus einem einzelnen Beschreibungssatz in natürlicher Sprache sowie verschiedenen formalen und inhaltlichen Zusatzangaben (vgl. Ebert 2012, 96f.). Der Beschreibungssatz ist der inhaltliche Kern der Anforderung, da in ihm die Anforderung ausformuliert ist. Die Formulierung dieses Satzes muss deutlich und klar sein; die Grundstruktur ist immer gleich (vgl. Ebert 2012, 92f.):

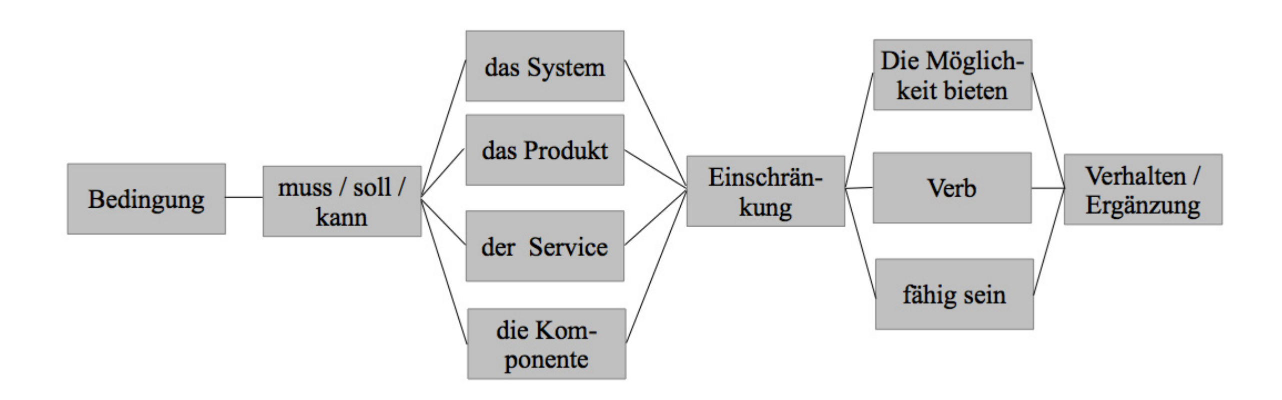


Abbildung 1: Vorlage für die Beschreibung einer einzelnen Anforderung
Quelle: in Anlehnung an Ebert 2012, 92

Der Beschreibungssatz sagt damit aus, unter welcher Bedingung das System etwas Bestimmtes leisten soll. In der Wahl des Hilfsverbs drückt sich die Priorität der

Anforderung aus: kann-Anforderungen haben geringe Priorität, soll-Anforderungen sind wünschenswerte Ziele und muss-Anforderungen sind verpflichtend (vgl. Rupp/SOPHIST GROUP 2009, 168f.). Damit alle am Entwicklungsprozess beteiligten Parteien ohne Missverständnisse mit einem Anforderungskatalog arbeiten können, empfiehlt sich das Anlegen einer Prozesswortliste. In ihr wird beschrieben, was die Verben und Begriffe, die in den Anforderungen auftauchen, konkret bedeuten. Beispiele für missverständliche Verben sind u. a. „eingeben“ oder „einfügen“ (vgl. Rupp/SOPHIST GROUP 2009, 169).

Der vollständige Katalog an Anforderungen für ein Projekt wird auch Anforderungsspezifikation genannt (Ebert 2012, 94f.). Neben dem Beschreibungssatz enthalten Anforderungen verschiedene Zusatzangaben; zu diesen gehören u.a.

- eine **Anforderungsnummer** zur eindeutigen Identifizierung,
- ein **Anforderungstitel** zur eindeutigen Beschreibung,
- der **Erfüllungsstatus**, um den Stand der Realisierung festzuhalten,
- sowie **Erläuterungen**, **Randbemerkungen**, **Querbezüge** oder **Kommentare** (vgl. Ebert 2012, 96f.).

Ein Anforderungsmuster enthält teilweise ähnliche Elemente wie eine Anforderung, hat aber die Aufgabe, das Schreiben der eigentlichen Anforderung zu erleichtern. Die Vorlagen für den Beschreibungstext, der den Kern der späteren Anforderung darstellt, ähneln dem Aufbau aus Abb. 2, jedoch sind einige der Textbausteine bereits vorgegeben. Zusätzlich zu diesen Elementen gibt ein Anforderungsmuster wichtige Hinweise dafür, wann das Muster anzuwenden ist, wie man Anforderungen damit schreibt und ggf. auch, wie diese Anforderungen implementiert und getestet werden können (vgl. Withall 2007, 21). Typische Elemente eines Anforderungsmusters sind u.a.

- die **Nummer** und der **Titel** des Musters,
- **grundlegende Details** und formale Angaben,
- sein **Anwendungsbereich**,
- **Hinweise zum Formulieren**,
- der **Inhalt** der Anforderung,
- sowie **Beispiele** und **Ergänzungen** (vgl. Withall 2007, 21ff.).

Der genaue Aufbau der hier erstellten Anforderungsmuster folgt den von Franch et al. (2010, 87f.) vorgeschlagenen Elementen und der von Hoffmann, Hoffmann und Leimeister vorgeschlagenen Struktur (2012, 387f.). Er setzt sich aus Metadaten und Vorlagen zusammen. Die Metadaten dienen dazu, das Muster inhaltlich einzuordnen:

Attribut	Funktion innerhalb des Anforderungsmusters
Nummer	Eindeutige Nummer zur Identifizierung des Musters.
Titel	Prägnanter, aussagekräftiger Name des Musters.
Ziel	Das Ziel, welches das System bei Berücksichtigung der Anforderung erfüllt. Das Ziel ist eine Entscheidungshilfe zur Nutzung oder Nichtnutzung des Musters.
Grundlage	Die theoretischen, fachlichen, rechtlichen oder sonstigen Hintergründe des Musters.
Abhängigkeiten	Anforderungsmuster, die in Kombination mit diesem Muster umgesetzt werden sollten.
Verknüpfungen	Anforderungsmuster mit einem ähnlichen Ziel.
Konflikte	Anforderungsmuster, deren Ziele dem aktuellen Ziel entgegenstehen.

Tabelle 1: Metadaten des Anforderungsmusters
Quelle: Hoffmann/Hoffmann/Leimeister 2012, 387

Die Vorlagen helfen dabei, die eigentlichen Anforderungen zu schreiben:

Attribut	Funktion innerhalb des Anforderungsmusters
Standardisierte Anforderung	Standardisierter Beschreibungssatz, der unverändert als Anforderung verwendet werden kann.
Erweiterung	Um diverse Parameter erweitertes Template, das detailliertere und speziellere Anforderungen erlaubt.
Parameter	Variable Teile des erweiterten Templates, die bestimmte Werte haben können.
Werte	Elemente, Wörter oder Phrasen, die als Parameter in das erweiterte Template eingebaut werden können.
Beispiele	Konkrete Beispiele, wie das Muster bereits genutzt wurde.
Hinweis	Wenn nötig, werden hier notwendige Handlungsimplikationen und Hinweise angegeben.

Tabelle 2: Attribute der Vorlage des Anforderungsmusters
Quelle: Hoffmann/Hoffmann/Leimeister 2012, 388

Es sei angemerkt, dass die vorliegende Arbeit nicht alle möglichen Detailangaben eines Anforderungsmusters verwendet, sondern nur jene berücksichtigt, die sich für eine von speziellen Systemen losgelöste Betrachtung eignen.

2.2 Usability

Für den Begriff „Usability“ gibt es verschiedene Definitionen, die nicht nur aus der Fachliteratur stammen, sondern auch aus offiziellen Standards für Produkt- oder Softwareentwicklung. Usability (auch: Benutzbarkeit, Gebrauchstauglichkeit) beschreibt, wie gut ein System oder in diesem Fall eine Software für ihre Anwender zu benutzen ist (vgl. Ebert 2012, 77f.). Die ISO-Norm 9241-11 definiert Usability wie folgt:

> „Das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen.“ (International Standards Organization 2006)

Die ISO/IEC 9126-1, ein Standard, der die Merkmale von Software-Qualität betrachtet, definiert Usability folgendermaßen:

> „Usability: the capability of the software to be understood, learned, used and attractive to the user, when used under specified conditions.“ (International Standards Organization 2001, zitiert nach Bevan 2001, 537)

Diese Definition berücksichtigt, das Usability zum Ziel einer Software beiträgt, die Nutzererwartung zu erfüllen (Bevan 2001, 537). Außerdem wird Lernbarkeit genannt, die bei vielen Autoren als wichtiges Usability-Merkmal gesehen wird (vgl. Nielsen 1993, 27f.; Benyon 2010, 84; Ludewig/Lichter 2010, 371). Andere oft genannte Usability-Merkmale sind eine geringe Anzahl an Fehlern während der Nutzung, eine gute Merkbarkeit der Funktionsweisen des Systems sowie die kognitive Belastung des Nutzers: gerade bei mobilen Anwendungen, die oft parallel zu anderen Aktivitäten genutzt werden, überbeanspruchen Systeme mit guter Usability ihre Anwender nicht (vgl. Harrison/Flood/Duce 2013, 4f.).

In beiden genannten Definitionen spielt der Nutzungskontext eine Rolle: es ist grundsätzlich zu beachten, dass Usability keine integrierte Systemeigenschaft ist, sondern ein Potenzial, das sich in Kombination mit verschiedenen Faktoren des Nutzungskontextes ergibt. Dies sind v.a. der Benutzer (sein Wissen, sein Können,

etc.), die Nutzungssituation und die Komplexität der zu bewältigenden Aufgabe[1]. Beispielsweise wird ein geübter Programmierer anders mit einer bestimmten Benutzeroberfläche umgehen als eine Person, die nicht computeraffin ist.

Die vielen existierenden Definitionen und Sichtweisen guter Usability deuten auf die große Bedeutung dieses Themas hin. Gerade dann, wenn die Nutzerakzeptanz einer Software für den Markterfolg einer Dienstleistung entscheidend ist – z. B. bei der Verbreitung einer Smartphone-Applikation oder einer Software für den Heim-PC – ist gute Usability ein entscheidender Erfolgsfaktor (vgl. Ludewig/Lichter 2010, 370f.). In der Fach- und Praxisliteratur haben sich daher unzählige, mehr oder weniger konkrete Empfehlungen herausgebildet, bei deren Beachtung man von guter Usability spricht.

Werden Usability-Anforderungen an ein System gestellt, gelten diese laut gängiger Anforderungsliteratur als Qualitätsanforderungen (vgl. z. B. Ebert 2012, 73; Withall 2007, 18). Gute Usability ergibt sich jedoch auch aus der Funktionalität eines Systems. Mathieson und Keil beispielsweise zeigen, dass Benutzeroberflächen nicht allein für die wahrgenommene Einfachheit der Nutzung verantwortlich sind, sondern auch die dahinter verborgene Funktionalität (1998, 227).

Obwohl das Überprüfen von Qualitätsanforderungen als schwierig gilt (vgl. Ebert 2012, 72ff.), hat sich für den Bereich der Usability eine eigene Arbeitsdisziplin, die sog. „Usability-Evaluation“, herausgebildet. Hierbei geht es darum, Systeme mit Hilfe verschiedener Methoden auf ihre Benutzbarkeit hin zu testen (vgl. Nielsen 1994, 1ff.). Es wird empfohlen, Usability Evaluierung möglichst früh im Entwicklungsprozess einzusetzen, da viele der Systemkomponenten, die für gute Benutzbarkeit zuständig sind, nur schwer änderbar sind, sobald das System einen gewissen Reifegrad erreicht hat (vgl. Holzinger 2005, 72). Hinzu kommt, dass es für Entwickler und Gestalter oft schwierig ist, abzuschätzen, an welchen Stellen die unterschiedlichen Nutzergruppen eines Systems bei der Nutzung Probleme bekommen könnten und deshalb frühzeitig deren Perspektiven berücksichtigt werden sollten (vgl. Harty 2011, 45f.). Um das Berücksichtigen von Usability besser in die Abläufe der Softwareentwicklung einzubinden, hat sich das Arbeitsfeld des Usability Engineerings entwickelt. Diese Form der Softwareentwicklung bezieht die Endnutzer des zu schaffenden Systems von

[1] vgl. Harrison/Flood/Duce 2013, 3f.; Hertzum 2010, 569ff.; Bevan 2001, 536f.; Nassar 2012, 1054.

Anfang an in die Entwicklung mit ein. Dies geschah zunächst durch frühzeitige Nutzungstests, mittlerweile arbeitet Usability Engineering aber auch verstärkt mit Anforderungen (vgl. Rosson/Carrol 2002, 14f.).

Der genannte Fokus auf die Endnutzer und deren Perspektive(n) ist für das Designen von Mensch-Maschine-Interaktion unentbehrlich. Ein weiteres zu nennendes Arbeitsfeld ist das sog. „User Centered Design". In dieser Design-Disziplin wird versucht, kompromisslos für die Perspektive des Endnutzers zu gestalten. Interaktionen sollen so gestaltet werden, dass sie für den Endnutzer möglichst einfach und logisch sind, auch wenn dies bedeutet, dass das zu schaffende System für Designer und Programmierer erheblich schwerer zu erstellen ist (vgl. Hix/Hartson 1993, 29f.). Zwingende Grundvoraussetzung für das Erarbeiten eines Software-Projektes ist daher, die Endnutzer und ihre zu lösenden Aufgaben genau zu verstehen (vgl. Benyon 2010, 84f.; Hix/Hartson 1993, 29f.). Das gilt auch, wenn Usability-Empfehlungen in Form von Anforderungen in den Entwicklungsprozess aufgenommen werden. Der in dieser Arbeit vorgestellte Katalog an Anforderungsmustern soll als Hilfsmittel für einen Anforderungsersteller verstanden werden, der seine Endnutzer und deren Aufgaben versteht und mit diesem Wissen Anforderungen zusammenstellt.

Bei dem Vergleich verschiedener Sichtweisen auf Usability verschwimmt mitunter die Grenze zwischen den Merkmalen guter Benutzbarkeit und Handlungsempfehlungen zur Schaffung von eben dieser; entsprechende Überschneidungen werden in den folgenden Kapiteln näher betrachtet. Die vorliegende Arbeit konzentriert sich auf die folgenden Kernmerkmale guter Usability (Harrison/Flood/Duce 2013, 4f.):

- **Effektivität der Nutzung:** ist die Fähigkeit des Nutzers, bestimmte Aufgaben mit Hilfe des Systems zu erfüllen.
- **Effizienz der Nutzung:** ist die Fähigkeit des Nutzers, seine Aufgaben schnell und präzise zu erledigen.
- **Zufriedenstellung**: ist der wahrgenommene Grad an Komfort und Behaglichkeit während der Nutzung.
- **Lernbarkeit der Nutzung eines Systems:** ist die Einfachheit, mit der Anwender eines Systems Professionalität in der Nutzung erreichen.
- **Merkbarkeit der Systemfunktionsweise:** ist die Fähigkeit des Nutzers, einen gewissen Grad an Professionalität der Nutzung auch nach längerer Inaktivität beizubehalten.

- **Geringe Fehlerraten:** beziehen sich auf eine möglichst geringe Zahl an Fehlern während der Nutzung.
- **Geringe kognitive Belastung:** als geringe Anzahl kognitiver Prozesse während der Nutzung, da davon ausgegangen werden muss, dass viele Applikationen parallel zu anderen Aktivitäten genutzt werden.

Alle im Folgenden genannten Handlungsempfehlungen und Anforderungsmuster dienen dazu, mindestens eines dieser Usability-Erfolgsmerkmale zu unterstützen.

3 Forschungsmethodisches Vorgehen

Im folgenden Abschnitt wird das forschungsmethodische Vorgehen der Arbeit beschrieben. Es teilt sich auf in das Vorgehen bei der Literaturrecherche, das Vorgehen bei der Expertenbefragung und das Vorgehen bei der praktischen Anwendung der Muster an einem Prototyp.

3.1 Vorgehen bei der Literaturrecherche

Um eine möglichst breite Sammlung an Handlungsempfehlungen für gute Usability zu gewinnen, wurde eine Literaturrecherche durchgeführt. Usability wird vorrangig in den Disziplinen der Softwareentwicklung, des Produktdesigns, des Interfacedesigns und der Software-Ergonomie behandelt. Ziel der Recherche war es, aus möglichst all diesen Bereichen Gestaltungshinweise zu bekommen, um daraus Anforderungsmuster ableiten zu können. Im Folgenden wird das Vorgehen der Literaturrecherche genauer beschrieben. Das Vorgehen folgt der systematischen Literatursuche von Kitchenham et al. (2009).

Im ersten Schritt werden Forschungsfragen formuliert, mit denen relevante Literatur gefunden werden soll (Brereton et al. 2009, 575; Kitchenham et al. 2009, 8). Einerseits werden Handlungsempfehlungen für das Schaffen guter Usability gesucht. Andererseits sollen aus diesen Handlungsempfehlungen Anforderungen und Anforderungsmuster abgeleitet werden. Da Anforderungen wie Ziele gehandhabt werden, müssen sie konkret und messbar ausformuliert sein (Ebert 2012, 94). Um die analysierte Literatur auszuwerten, wird die folgende Forschungsfrage definiert:

1. Welche Software-Anforderungen für Usability gibt es und welche Anforderungsmuster lassen sich daraus ableiten?

Im zweiten Schritt werden Kriterien aufgelistet, mit deren Hilfe die betrachtete Literatur bewertet wird (Kitchenham et al. 2009, 8). Für die vorliegende Problemstellung ist Literatur aus dem Bereich der Usability notwendig, die konkrete Gestaltungsvorschläge für die Nutzeroberflächen und die Funktionen der Software gibt. Die Kriterien, die sich damit ergeben, lauten

- Die Literatur gibt Gestaltungsempfehlungen für Systeme mit guter Usability.

- Die Gestaltungsempfehlungen sind konkret, d.h. sie beschreiben mindestens eine Systemanforderung.
- Die Gestaltungsempfehlungen haben den Charakter von Anforderungen oder lassen es zu, Anforderungen abzuleiten. Gesucht werden keine reinen Lösungsvorschläge.
- Die Gestaltungsempfehlungen sind auf Software-Applikationen übertragbar.

Im nächsten Schritt sind relevante wissenschaftliche Quellen zu identifizieren (Kitchenham et al. 2009, 8f.). Gesucht wurde in den Online-Verzeichnissen Google Scholar sowie EBSCO: Business Source Premier. Google Scholar enthält ein breit gefächertes Angebot an Literatur aus den Bereichen Software Engineering sowie Design. EBSCO wird zusätzlich hinzugezogen, da hier u.a. Fachbeiträge aus dem Bereich der Betriebswirtschaftslehre enthalten sind. Mit diesen Verzeichnissen sollen sowohl technische als auch organisatorische Perspektiven auf Usability berücksichtigt werden. In diesen Suchmaschinen wurde mit folgenden Stichwörtern gesucht:

- „Usability“: Usability taucht in allen betrachteten Disziplinen – Usability Design, Softwareentwicklung, Interface Design und Software Ergonomie – als eigenes Handlungsfeld auf. Es wurde daher „Usability“ als fachübergreifendes Stichwort genutzt.
- „ease of use“: Usability wird oft mit leichter Benutzbarkeit gleichgesetzt. Für die Bereiche der Softwareentwicklung und des Product Designs wurde daher „ease of use“ als Stichwort verwendet.
- „software ergonomics“: Die Disziplin der Mensch-Maschine-Interaktion (auch: Human-Machine-Interaction, HMI) verwendet neben „Usability“ und „ease of use“ oft Ergonomie/ergonomics als Namen für das beschriebene Gestaltungsfeld, weshalb nach „software ergonomics“ gesucht wurde.

Google Scholar und Business Source Premier sortieren die Suchergebnisse nach Relevanz zum Suchbegriff. Nach ca. 70 Ergebnissen nimmt die tatsächliche, inhaltliche Relevanz zu den gesuchten Begriffen jedoch stark ab, da ab dort sehr spezielle Arbeiten in die Ergebnisse miteinfließen. Es wurden daher für jedes Stichwort in jeder Suchmaschine nur die 100 am besten bewerteten Ergebnisse berücksichtigt.

Schlussendlich wurden die genannten Quellen zusammengetragen und den genannten Kriterien entsprechend untersucht. Mit Hilfe der zuvor formulierten Forschungsfragen konnten jene Arbeiten erfasst werden, die zur erfolgreichen Ableitung von Usability-Anforderungsmustern geeignet sind.

Die durch die Literaturrecherche gefundenen Gestaltungsempfehlungen werden in den Kapiteln 4.1 – 4.15 diskutiert. Sofern aus den Empfehlungen sinnvolle, allgemeingültige Anforderungen abgeleitet werden können, werden diese in den Unterkapiteln in standardisierter Form angegeben. Der vollständige Katalog an Anforderungsmustern findet sich in Anhang A.

3.2 Vorgehen bei der Expertenbefragung

Die in Kapitel 4.1 erarbeiteten Usability-Anforderungsmuster sollen in Kapitel 5 auf ihre Alltagstauglichkeit hin untersucht werden. Dazu werden Interviews mit Software-Entwicklern geführt, die mit Hilfe von Anforderungen Software-Applikationen entwickeln.

Potentielle Unternehmen, in denen sich entsprechende Experten finden, wurden nach folgenden Kriterien ausgewählt:

1. Das Unternehmen entwickelt eigene Software-Applikationen und setzt sowohl in der internen Koordination als auch im Kundenkontakt Requirements Engineering ein.
2. Die entwickelte Software richtet sich an Endnutzer, die keine oder wenige Programmierkenntnisse besitzen. Solche Software macht das besondere Berücksichtigen von Usability erforderlich.

Den ausgewählten Interview-Partnern wurden zunächst die im Anhang aufgeführten Usability-Anforderungsmuster zur Verfügung gestellt. Danach wurden in Einzelgesprächen mit den Anforderungs-Ingenieuren der jeweiligen Unternehmen Anforderungsmuster diskutiert. Für diese Diskussion wurde der nachfolgende Fragenkatalog verwendet. Darin werden die Qualitätsmerkmale guter Anforderungsmuster abgefragt. Diese Kriterien beruhen vorrangig auf einem Qualitäts-Modell, das von Wurhofer et al. (2010) entwickelt wurde. In diesem Qualitäts-Modell werden fünf Bereiche identifiziert, die die Qualität eines Anforderungsmusters ausmachen: Auffindbarkeit, Verständlichkeit, Hilfestellung,

empirische Bestätigung und allgemeine Akzeptanz. Vor der Prüfung der Muster wurden außerdem einige allgemeine, unternehmensbezogene Themen angesprochen.

Zuerst sollte herausgefunden werden, wie das Anforderungsmanagement des jeweiligen Unternehmens abläuft. Dazu gehört u. a., welche Stakeholder mit den Anforderungen arbeiten, ob und wie Kunden einbezogen werden und wie konkret die Anforderungen ausfallen. Es wurde außerdem nach der durchschnittlichen Anzahl an Anforderungen eines Projektes gefragt, da sich daraus Rückschlüsse auf den Detailgrad der Anforderungen ergeben können (vgl. Ebert 2012, 99).

A. Allgemeine Fragen
A.a Arbeiten Sie bei jedem Projekt mit Anforderungen?
A.b Wie viele Anforderungen hat ein Projekt bei Ihnen im Durchschnitt?

Mit den folgenden Fragen werden die für die Diskussion relevanten Kenntnisse des Unternehmens erfragt.

A.c Arbeiten Sie mit Anforderungsmustern?
A.d Berücksichtigen Sie Usability in der Softwareentwicklung?
A.e. Verwenden Sie Usability-Anforderungen?

Zur Prüfung der Anforderungsmuster wurden die genannten Qualitätsbereiche herangezogen.

Auffindbarkeit beschäftigt sich mit der Frage, ob die Muster innerhalb ihrer Anordnung und Benennung gut gefunden werden können. Das Kriterium beruht auf der Annahme, dass Muster, die bei Bedarf schlecht zu finden sind, vermutlich nicht verwendet werden würden. Gute Auffindbarkeit kann an der Qualität der Gliederung oder Hierarchie der Muster abgelesen werden (vgl. Wurhofer et al. 2010, 255f.). Es ergaben sich folgende Interview-Fragen:

B. Auffindbarkeit
B.a Wie beurteilen Sie die Gliederung der Usability-Anforderungsmuster? Ist die Reihenfolge sinnvoll?
B.b Können Sie bestimmte Muster gut finden, wenn Sie an ein bestimmtes Problem denken?

Das Kriterium der Verständlichkeit betrachtet, ob Anwender die Muster verstehen. Die Verständlichkeit kann daran abgelesen werden, ob sich den Befragten der Sinn der Muster sowie deren Elemente erschließen. Dazu gehört auch die verwendete Sprache in den Mustern. Die Anforderungsmuster sollten vollständig sein und alle notwendigen Informationen enthalten, damit man sie effektiv verwenden kann. Sie sollten bestimmte Probleme behandeln und eine deutliche Problem-Lösungsbeziehung erkennen lassen. Es ist außerdem zu beachten, ob in den Anforderungsvorlagen die richtige Balance zwischen Konkretheit und Abstraktion gefunden wurde (vgl. Petter/Khazanchi/Murphi 2010, 18; Wurhofer et al. 2010, 256f.):

C. Verständlichkeit
C.a Sind alle Elemente der Muster verständlich?
C.b Sind alle notwendigen Informationen enthalten?
C.c Sind die Muster verständlich geschrieben/formuliert?
C.d Sind die Muster ausreichend problembezogen?
C.e Ist die Balance zwischen Konkretheit und Abstraktion angemessen?

Es wurde außerdem erfragt, ob die erarbeiteten Anforderungsmuster eine ausreichende Hilfestellung sein könnten. Dazu gehört, ob das Erstellen von Anforderungen mit den Mustern überhaupt durchführbar ist, ob Anwender also genug Informationen für die tatsächliche Bearbeitung erhalten. Es ist zu prüfen, ob Designs tatsächlich durch die Muster verbessert werden können, ob sie bei der Lösung bestimmter Probleme helfen und ob sie eine gemeinsame Wissensbasis für die Entwickler sein können (Wurhofer et al. 2010, 257; Petter/Khazanchi/Murphi 2010, 19f.):

D. Hilfestellung
D.a Können die Muster/Anforderungen Ihnen dabei helfen, ein Design zu verbessern?
D.b Helfen die Muster bei der Problemlösung? Sind typische Usability-Probleme abgedeckt?
D.c Schaffen die Muster eine gemeinsame Basis für die Kommunikation? Entspricht die Sprache der Ihres Teams?
D.d Enthalten die Muster für Sie relevantes Wissen?
D.e Kann man die Muster gut anwenden? Was würden Sie verändern, um Sie in Ihre Abläufe einzubringen?

Das Qualitätsmodell von Wurhofer et al. nennt als viertes Qualitätskriterium, dass die Muster empirisch belegt sein müssen. Mit diesem Kriterium soll geprüft werden, ob die Muster aus empirischen Studien heraus entstanden sind (2010, 257f.). Dieses Kriterium wurde in den Expertengesprächen nicht gesondert abgefragt, da die Muster bereits auf der Grundlage einer Fachliteratur-Recherche entstanden sind.

Das letzte Qualitätskriterium beschäftigt sich mit der allgemeinen Akzeptanz der Muster. Dabei wird gefragt, ob Anwender mit den Mustern inhaltlich einverstanden sind und ob sie an den Nutzen der Muster glauben. Es wird angenommen, dass Anwender Muster, an die sie nicht glauben, nicht verwenden werden. Es ergeben sich die folgenden, abschließenden Fragen:

E. Allgemeine Akzeptanz
E.a Stimmen Sie mit den Mustern inhaltlich überein oder haben Sie andere Erfahrungen oder Kenntnisse?
E.b Glauben Sie an die Nützlichkeit dieser Muster?

Die Ergebnisse der Befragung finden sich zusammengefasst in Kapitel 5. Darin werden sowohl die befragten Unternehmen beschrieben, als auch die gegebenen Antworten gegenübergestellt. Änderungen an den Usability-Anforderungsmustern, die sich aus den Gesprächen ergaben, sind in dem endgültigen Musterkatalog und den Ergebnissen aus Kapitel 4 bereits enthalten.

3.3 Vorgehen bei der praktischen Anwendung der Muster an einem Fallbeispiel

In Kapitel 6 wird der Katalog aus Usability-Anforderungsmustern an einem Fallbeispiel angewendet. Dieses Fallbeispiel ist ein unvollständiger Prototyp einer Smartphone-Applikation, kurz App. Für diesen Prototypen sind der Zweck und die beabsichtigte Funktionalität definiert. Es fehlen jedoch einige Bildschirmausschnitte und Dialoge, die noch zu entwerfen sind.

Zur praktischen Anwendung der Usability-Anforderungsmuster werden zunächst die Applikation, ihr Zweck und ihre Zielgruppe beschrieben. Auf dieser Grundlage werden dann alle Anforderungsmuster des Katalogs auf ihre Anwendbarkeit hin geprüft und so weit wie möglich angewendet. Es entstehen dadurch Usability-Anforderungen an die Applikation. In einem dritten Schritt werden mit Hilfe der zuvor

erarbeiteten Gestaltungsempfehlungen Lösungsvorschläge erarbeitet, mit denen die Applikation den Usability-Anforderungen gerecht werden kann.

4 Usability-Empfehlungen in Form von Anforderungsmustern

Im folgenden Teil dieser Arbeit soll ein Katalog mit Anforderungsmustern für Usability erarbeitet werden. Mit diesem Katalog an Mustern soll es möglich sein, Usability-Anforderungen für typische Software-Applikation aufzustellen. Grundlage für die dort berücksichtigten Empfehlungen ist eine Literaturrecherche aus den Bereichen Usability für Software, Produktdesign, Interfacedesign und Software-Ergonomie. Die behandelten Usability-Bereiche befassen sich zunächst mit den Steuermöglichkeiten für den Nutzer (Kapitel 4.1 – 4.3), später mit den Informationen, die das System anbietet (Kapitel 4.4 – 4.10) und schlussendlich mit einigen schwerer greifbaren Designthemen (Kapitel 4.11 – 4.15). Dabei werden in jedem Unterkapitel der jeweilige Usability-Bereich vorgestellt und Handlungsempfehlungen zusammengetragen. Viele häufig genannte Empfehlungen fallen in den Bereich der Gestaltung und Problemlösung. In jenen Fällen wird versucht, aus den dahinter liegenden Problemstellungen Anforderungsmuster abzuleiten. Die vollständigen Anforderungsmuster sind dem Anhang zu entnehmen. Im Fließtext werden als Zwischenergebnisse die standardisierten Anforderungstexte und die dazugehörigen Nummern der Muster angegeben.

Ein wichtiges Qualitätskriterium für Anforderungen und somit auch für Anforderungsmuster ist deren Konsistenz: einzelne Anforderungen dürfen sich gegenseitig nicht widersprechen und müssen eine eindeutige Wort- und Begriffswahl haben, die im kompletten Anforderungskatalog durchgehalten wird (Pohl/Rupp 2009, 50). Zu diesem Zweck wird eine Prozesswortliste erstellt. Diese – nachfolgend dargestellt – definiert Begriffe, die innerhalb der Anforderungstexte auftauchen und ohne weitere Erläuterung in ihrer Bedeutung nicht eindeutig sein könnten:

Begriff	Bedeutung
Aktion	Das Auslösen einer (Teil-)Funktion des Systems durch Aktivieren eines Bedienelementes oder durch Tätigen einer Eingabe.
anbieten	Darstellung einer Information, eines Inhalt oder eines Bedienelements, die den Nutzer erkennen lässt, dass er sie/ihn/es benutzen kann und soll.
Applikation	Sammelbegriff für Software-Anwendungen, die von einem Menschen zur Aufgabenerfüllung genutzt werden.
Aufgabe	Die Gesamtheit der notwendigen Schritte, die der menschliche Nutzer zum Erreichen eines Zieles durchführen muss und für die er eine oder mehrere Funktionen der Applikation nutzt.
Bedienelemente	Zeichen, Bilder oder Eingabefelder, die das Durchführen einer Aktion ermöglichen.
Dialog	Eine oder mehrere Funktionen einer Applikation, bei denen der Nutzer mit der Applikation interagieren kann.
Eingabe	Das Eingeben von Werten, Zeichen (Buchstaben oder Zahlen), Bewegungen (Zeichnungen, etc.) und Geräuschen innerhalb eines Eingabefeldes oder Aufnahme-Eingangs.
Einstellungen	Bestimmte Kombinationen aus Eingaben innerhalb eines Einstellungsmenüs.
Fehler	Falsche Verwendung der Applikation, die zu Ergebnissen führt, die der Nutzer nicht wollte oder die zu gar keinem Ergebnis führt, weil die Applikation bestimmte Aktionen nicht erlaubt.
Funktion	Zusammenhängende Abfolge von Vorgängen der Applikation, durch die ein Ziel der Applikation erreicht wird.
Hilfestellung	Hinweise in Form von Text, Bildern oder Geräuschen, die dem Nutzer die richtige Verwendung der Applikation erklären.
Information	Alle Formen von Informationen aus dem Inhalt, der Navigation oder anderen Bestandteilen der Applikation, bestehend aus Text, Bildern, Bewegungen oder Geräuschen.
Inhalt	Von der Applikation dargestellte Objekte, die nicht ausschließlich zur Navigation gehören.
Navigation	Gesamtheit aller Bedienelemente, die das Aufrufen der verschiedenen Bereiche der Applikation ermöglichen.
Nutzer	Der menschliche Benutzer des Systems.
Optionen	Mögliche ausführbare Funktionen.
Weg	Mögliche, nutzbare Reihenfolge von Aktionen und Funktionen.

Tabelle 3: Prozesswortliste
Quelle: Eigene Darstellung

4.1 Kontrolle und Einschränkungen

Kontrolle und Einschränkungen sind ein oft genanntes Gestaltungsfeld für Usability. Im weitesten Sinne geht es hier um den Umfang an Kontrolle, den ein Nutzer über die Funktionen und Prozesse des Systems hat.

Eine oft anzutreffende Empfehlung lautet, Nutzern so viel Kontrolle wie möglich über ein System zu geben (vgl. Jordan 1998, 31ff.). Verschiedene Untersuchungen ergaben,

dass wahrgenommene Kontrolle die Nutzungsabsicht für ein System erhöhen kann (vgl. Venkatesh 2000, 346). Es besteht zunächst die Forderung nach grundlegenden Steuerungsmöglichkeiten eines Interfaces. Die ISO-Norm 9241 definiert Steuerbarkeit wie folgt (vgl. International Standard Organization 2006):

> „Ein Dialog ist steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist."

Es wird erwartet, dass Nutzer sich in einer Navigation vor- oder zurückbewegen sowie Aktionen rückgängig machen oder neustarten können (Nassar 2012, 1055). Steuerungsmöglichkeiten geben dem Nutzer ein Gefühl von Kontrolle und Sicherheit, sie müssen deshalb nicht nur vorliegen, sondern auch vom Nutzer wahrgenommen werden (Rampl 2007). Die Empfehlung wird von Scapin und Bastien fortgeführt (1997, 225): Computer sollten nur jene Vorgänge ausführen, die der Nutzer bewusst ausgelöst hat. Unter dieser Voraussetzung seien Nutzer in der Lage, die Funktionsweise der Software zu verstehen und zu lernen, gleichzeitig werde die Fehlerhäufigkeit reduziert. Die Autoren unterscheiden zwei Formen der Kontrolle: Aktionskontrolle und Nutzerkontrolle. Aktionskontrolle ist die Beziehung zwischen den Aktionen des Nutzers und den Prozessen innerhalb der Software. Zwischen beidem sollte eine explizite Beziehung bestehen, d. h. das System sollte nur Vorgänge ausführen, die der Nutzer ausgelöst hat. Nutzerkontrolle ist die Möglichkeit des Nutzers, die Vorgänge des Systems zu steuern, d. h. sie auszulösen, sie zu unterbrechen, sie abzubrechen, etc. Dies trägt ebenfalls zu einer geringeren Fehlerquote bei und macht das System für Nutzer berechenbarer (vgl. Scapin/Bastien 1997, 225). Diese Empfehlungen führen zu den folgenden standardisierten Anforderungen:

KO-01: Explizite Aktionskontrolle
Die Applikation soll nur die Funktionen ausführen, die der Nutzer mit seinen Aktionen ausgelöst hat.

KO-02: Steuerbarkeit durch Vorliegen einer Navigation
Die Applikation soll dem Nutzer ermöglichen, sich innerhalb der Dialogabfolge vor- und zurückzubewegen.

KO-03: Funktionen pausieren und abbrechen
Während die Applikation eine Funktion ausführt, soll der Nutzer die Möglichkeit haben, die Funktion zu unterbrechen und sie später fortzusetzen.

Während die Applikation eine Funktion ausführt, soll der Nutzer die Möglichkeit haben, die Funktion abzubrechen.

Die Forderung nach voller Kontrolle für den Nutzer steht dem Anbieten von Bequemlichkeit entgegen. Gerade bei Software-Systemen werden typischerweise bestimmte Einstellungen oder Eingaben automatisiert vorgenommen, um dem Nutzer die Eingabe zu ersparen. Ein Beispiel sind die Ausgangseinstellungen eines Schreibprogrammes, bei dem während des Öffnens eines neuen Dokumentes bereits Seitenränder, Schriftgröße oder Zeilenabstand voreingestellt sind. Hier ist es wichtig, den Nutzer darauf hinzuweisen, dass das System eine Ausgangseinstellung vorgenommen hat. Bei dem genannten Beispiel eines Schreibprogramms geschieht dies mitunter durch den permanenten Hinweis auf die zurzeit genutzte Formatvorlage (vgl. Jordan 1998, 32). Es gibt außerdem die Empfehlung, den Nutzer eigene Voreinstellungen speichern zu lassen, die bei Aufruf eines Eingabeformulars aufgerufen werden (Knittle/Ruth/Gardner 1987, 171). Entsprechende Anforderungen können lauten:

KO-04: Information zu Vor-Einstellungen
Wenn die Applikation in einem Einstellungsmenü selbstständig Einstellungen vornimmt, soll die Applikation den Nutzer über diese Vor-Einstellung informieren.

KO-05: Vor-Einstellungen abspeichern
Wenn ein Einstellungsmenü auszufüllen ist, soll die Applikationen dem Nutzer die Möglichkeit bieten, Einstellungen abzuspeichern und wiederzuverwenden.

Der Forderung nach voller Kontrolle für den Nutzer steht nicht nur die Forderung nach Bequemlichkeit entgegen, sondern auch das Berücksichtigen der Nutzerkenntnisse und die Vermeidung von Fehlern (vgl. Lidwell/Holden/Butler 2003, 64). Denkt man an Situationen, in denen ein Anwender ein kompliziertes System zum ersten Mal benutzt, erscheint es bedenklich, ihm sofort eine Bedienoberfläche mit der vollen Systemkontrolle zu übergeben. Der Umfang an Funktionen könnte ihn überfordern

und Fehler erzeugen. Eine Methode, dies zu vermeiden, ist das Anbieten mehrerer möglicher Wege, eine Aufgabe durchzuführen: einen einfachen Weg mit eingeschränktem Einstellungsumfang und eine ausführliche Alternative mit allen Einstellungen. Beide Wege führen zum selben Ziel, der einfache Weg hat jedoch mehr Struktur und nimmt den Nutzer an die Hand, während der komplizierte Weg volle Flexibilität und Kontrolle bietet. Da mehrere Wege der Aufgabenlösung ein System kompliziert machen, sollte es maximal zwei Wege geben. So könnten die Bedürfnisse von Anfängern und Experten des Systems befriedigt werden (vgl. Lidwell/Holden/Butler 2003, 64; Stapelkamp 2010, 308f.; Borenstein/Thyberg 1991, 239f.). Ein solches Vorgehen empfiehlt sich bei komplexen Systemen. Es sind auch Applikationen denkbar, die so oft benutzt werden, dass Nutzer zwangsläufig Nutzungsexpertise erlangen (z. B. Geldautomaten). Solche Systeme sollten ebenfalls unter der Annahme gestaltet werden, dass alle Nutzer Erstanwender sind, aber nur einen Weg anbieten (vgl. Lidwell/Holden/Butler 2003, 64).

> **KO-06: Zwei Wege zur Aufgabenbewältigung**
> Wenn eine Funktion Einstellungen erlaubt, soll die Applikation dem Nutzer die Funktion einmal mit Vor-Einstellungen und einmal mit allen Einstellungsmöglichkeiten anbieten.

Zur Schaffung übersichtlicher Designs wird grundsätzlich empfohlen, keine überflüssigen Informationen anzuzeigen. In diesem Rahmen können Funktionen, die zu einem bestimmten Zeitpunkt während der Nutzung nicht verfügbar sind, ausgeblendet oder ausgegraut werden. Der Nutzer betrachtet die Funktion dann von vornherein nicht als mögliche Option (vgl. Lidwell/Holden/Butler 2003, 60).

> **KO-07: Kein Anbieten nicht verfügbarer Aktionen**
> Ist eine Funktion nicht ausführbar, soll die Applikation die Funktion nicht anbieten.

Das gezielte Einschränken von Funktionalität gilt teilweise als eigenes Gestaltungsfeld im Interface Design. Dabei wird beispielsweise zwischen physischen und psychischen Einschränkungen unterschieden (vgl. Bennet/Flach 2011, 112; Lidwell/Holden/Butler 2003, 60). Die hier möglichen Gestaltungsvarianten sind für eine Vielzahl an Softwareapplikationen geeignet, haben jedoch den Charakter reiner Lösungsvarianten und werden hier nicht weiter behandelt.

4.2 Effizienz

Effizienz gilt als eines der Kernmerkmale guter Benutzbarkeit. Anwender sollen bei Verwendung einer Software ihre Aufgaben mit dem geringstmöglichen Bedienaufwand erledigen können (vgl. Ludewig/Lichter 2010, 371). Um Effizienz der Bedienung zu erreichen, muss die Bedienoberfläche von jeder Art unnötiger Komplexität befreit werden. Grundsätzlich sollen alle Aktivitäten so gestaltet werden, dass sie nur minimalen körperlichen und geistigen Aufwand des Nutzers benötigen und diesen nicht ermüden (Connel et al. 1997, 34f.). Kompakt gestaltete Bedienoberflächen tragen nicht nur zu einer effizienten Nutzung, sondern auch zu einer geringeren Fehlerwahrscheinlichkeit bei (vgl. Scapin/Bastien 1997, 224). Die Forderung nach Effizienz steht vor dem Hintergrund, dass Nutzer Handlungen nur ausführen, wenn deren Aufwand geringer ist als deren Nutzen. Die bequeme, schnelle Nutzung einer Software ist damit ein zentrales Qualitätsmerkmal (vgl. Lidwell/Holden/Butler 2003, 68).

Die erste zu nennende Empfehlung betrachtet den durch das System dargestellten Inhalt. In den meisten Systemen benutzt ein Nutzer die Bedienoberfläche, um bestimmte Inhalte zu erreichen. Dabei ist zu beachten, dass auch die Bedienelemente Inhalte darstellen. Genau wie der eigentliche Inhalt beanspruchen die Bedienelemente Platz auf dem Bildschirm und beanspruchen die Aufmerksamkeit des Nutzers. Sind sie zu auffällig, können sie den Nutzer ablenken und die Konzentration erschweren. Die Bedienelemente sind daher so zu gestalten, dass sie den eigentlichen Inhalt in den Vordergrund stellen. Im besten Fall kann ein Nutzer direkt mit dem Inhalt interagieren, ohne dass eine Bedienoberfläche zwischengeschaltet oder losgelöst neben dem Inhalt angebracht ist (Blair-Early/Zender 2008, 102).

> **EF-01: Inhalt herausstellen**
> Hat eine Funktion den Zweck, Inhalt darzustellen, sollen die Bedienelemente nicht von diesem Inhalt ablenken.

Effizienz befasst sich zu großen Teilen mit den eigentlichen Funktionen des Systems. Grundsätzlich gilt, dass Nutzer eine Aufgabe mit so wenigen Aktionen wie möglich durchführen können sollen, damit Schnelligkeit und Effizienz erreicht werden (vgl. Lin/Choong/Salvendy 1997, 271). Der Aufwand des Anwenders ist zu minimieren und er sollte nur jene Aktionen durchführen, die essentiell für die Aufgabenerfüllung sind und nicht vom System selbst ausgeführt werden können. Dazu gehört auch, dass

Eingaben und Befehle möglichst bequem ausgeführt werden können und dass in der Vergangenheit erledigte Aufgaben kein zweites Mal ausgeführt werden müssen (Knittle/Ruth/Gardner 1987, 164).

EF-02: Aufwand des Nutzers minimieren
Der Nutzer soll nur Aktionen ausführen müssen, die seine Entscheidung erfordern.

EF-03: Körperlichen Einsatz minimieren
Das Durchführen von Aktionen durch den Nutzer soll mit dem geringstmöglichen körperlichen Einsatz erfolgen.

Die Aufgaben, die mit Hilfe einer Software erledigt werden sollen, sind oft sehr komplex. Durch eine einfache, klare Wortwahl und das Verwenden von Metaphern (siehe Kapitel 4.12 & 4.14) kann diese Komplexität verringert werden. Um längere Aufgaben für den Nutzer verständlich zu machen, wird empfohlen, komplexe Aufgaben in mehrere überschaubare Teilschritte aufzuteilen. Dadurch wird die Aufgabenerfüllung für den Nutzer geradliniger (vgl. Hix/Hartson 1993, 35).

EF-04: Verständliche Anweisungen
Wenn der Nutzer eine oder mehrere Aktionen durchführen soll, soll die Applikation Bildanweisungen mit klaren, unmissverständlichen Bildern und Symbolen verwenden.

Wenn der Nutzer eine oder mehrere Aktionen durchführen soll, soll die Applikation Textanweisungen mit kurzen, klaren Sätzen verwenden.

EF-05: Aufteilen komplexer Aufgaben
Die für die Durchführung der Aufgabe notwendige Aktionsabfolge soll in mehrere Teilschritte unterteilt werden.

Neben den körperlichen Anstrengungen einer Arbeit ist die geistige Anstrengung zu beachten. Das Design einer Benutzeroberfläche muss die begrenzte Kapazität des menschlichen Kurzzeitgedächtnisses berücksichtigen. Dazu wird empfohlen, die Informationsdichte, der sich ein Anwender während der Nutzung gegenüber sieht, gering zu halten. Es sollte außerdem vermieden werden, dass Nutzer sich Informationen von einem Zwischenschritt bis zum nächsten merken müssen. Je weniger Informationen der Nutzer verarbeiten muss, desto schneller kann er seine

Aufgabe erledigen und desto weniger Fehler macht er (vgl. Scapin/Bastien 1997, 224f.). Hinzu kommt, dass ein Anwender das System schneller zu benutzen lernt, wenn er sich weniger merken muss. Wenn Aktionen immer ähnlich ablaufen, vermindert dies ebenfalls die kognitive Belastung (vgl. Knittle/Ruth/Gardner 1987, 165; Lin/Choong/Salvendy1997, 271). Daher sind auch die in Kapitel 4.8 beschriebenen Anforderungen zu konsistenter Darstellung in diesem Rahmen zu erwähnen.

EF-08: Übersichtlichkeit
Wenn der Nutzer eine oder mehrere Aktionen durchführen soll, soll die Applikation nur Informationen anzeigen, die zur aktuell ausgewählten Funktion gehören.

EF-09: Kein Merken-Müssen von Informationen
Wenn eine Funktion mit mehreren Aktionen ausgeführt wird, soll die Applikation dem Nutzer stets alle notwendigen Informationen anzeigen.

Weiterhin sollen sich Nutzer nur Schritte merken müssen, die für die Aufgabenerfüllung essentiell sind. Es muss bei jeder Software abgewogen werden, welche Informationen wichtig sind und welche nicht. Das Merken und Lernen der Softwarefunktionen wird erhöht, wenn dem Nutzer erst nach und nach zusätzliche Optionen und Möglichkeiten vom System aufgezeigt werden (vgl. Knittle/Ruth/Gardner 1987, 165). Daher sind auch hier die Anforderungen der eingeschränkten Kontrolle aus Kapitel 4.1 zu nennen.

4.3 Individualisierbarkeit

Im vorletzten Kapitel deutete sich bereits an, dass das Anbieten von mehreren Wegen zur Lösung einer Aufgabe mit einer Software sinnvoll sein kann. Neben den grundsätzlichen Steuerungsmöglichkeiten wird die Individualisierbarkeit einer Software als Usability-Faktor diskutiert. Dabei geht es um das Potenzial der Software, sich an die körperlichen, geistigen und technischen Voraussetzungen des Nutzers sowie an seine Vorlieben anpassen zu können (vgl. Rampl 2007).

Die Grundlage für Individualisierbarkeit ist Flexibilität, d. h. das System muss verschiedene Funktionen, Wege oder sonstige Steuerungsmöglichkeiten zur

Aufgabenerfüllung anbieten. Die ISO-Norm 9241 definiert Individualisierbarkeit wie folgt:

> „Ein Dialog ist individualisierbar, wenn das Dialogsystem Anpassungen an die Erfordernisse der Arbeitsaufgabe, individuelle Vorlieben des Benutzers und Benutzerfähigkeiten zulässt." (International Standards Organization 2006)

Individualisierbarkeit wird im Rahmen von Usability diskutiert, weil ein System oder eine Software sehr unterschiedliche Nutzer haben kann. Anwender unterscheiden sich oft nicht nur nach ihrem Kompetenzgrad, sondern auch danach, wie sie Entscheidungen bei der Navigation treffen. Einige Anwender lassen sich von inhaltlichen, andere von ästhetischen oder emotionalen Faktoren leiten. Es gilt als große Herausforderung, eine Bedienoberfläche so zu gestalten, dass alle in Frage kommenden Nutzertypen zufrieden sind (vgl. Stapelkamp 2010, 308f.). Werden mehrere Möglichkeiten zur Aufgabenlösung angeboten, können damit mehr Menschen mit unterschiedlichen Fähigkeiten erreicht werden. Das führt zu einer größeren Zugänglichkeit der Software (Connel et al. 1997, 34; Lin/Choong/Salvendy 1997, 271). Werden außerdem verschiedene Wege zur Aufgabenlösung entsprechend den unterschiedlichen Vorlieben des Nutzers angeboten, wird dies allgemein als Beitrag zu guter Usability gewertet (vgl. Nassar 2012, 1055). Nutzer bevorzugen es grundsätzlich, bei Bedienoberflächen die Wahl zwischen verschiedenen Alternativen zu haben. Dies steigert ihre wahrgenommene Freiheit sowie die wahrgenommene Einfachheit der Nutzung. Die bevorzugte Alternative erzeugt außerdem ein Gefühl der Verbundenheit gegenüber der Funktion (vgl. Murray/Häubl 2010, 971ff.). Bietet ein System mehrere Navigationsformen, ist es allerdings auch komplexer: je mehr Wege und Optionen zur Aufgabenerfüllung bereitstehen, desto schwieriger kann es für Nutzer sein, sich in der Benutzeroberfläche zurechtzufinden und sie zu lernen. Hohe Flexibilität steht damit den Forderungen nach Effizienz und niedriger Komplexität entgegen (vgl. Lidwell/Holden/Butler 2003, 102; Stapelkamp 2010, 309).

Es bleibt die Frage, wie dem Spannungsfeld zwischen Flexibilität und Komplexität begegnet werden kann. Einerseits ist die Komplexität der betrachteten Applikation ausschlaggebend.

Nicht bei jeder Software-Applikation ist Individualisierbarkeit notwendig. Andererseits wird – ähnlich wie im vorangegangenen Kapitel – empfohlen, den Grad

an Flexibilität und Individualisierbarkeit an den Erfahrungsstand des Nutzers anzupassen.

Einfache Webseiten oder Smartphone-Applikationen kommen mit ebenso einfachen Bedienoberflächen aus. Hier bedeutet Individualisierung lediglich, dass die Anwendung die individuellen Einstellungen des Endgerätes, des Betriebssystems oder des Browsers akzeptiert (vgl. Rampl 2007). Eine entsprechende Anforderung ist CO-04 in Kapitel 4.8. Enthält eine Systemfunktion viel Nutzerinteraktion in Form von Dialogen und Aktionen, die sehr häufig ausgeführt werden, sollte das System persönliche Einstellungen anbieten (vgl. Rampl 2007; Knittle/Ruth/Gardner 1987, 171). Gerade die oft genutzten Funktionen sollten so gestaltet sein, dass Nutzer sie ihren Vorlieben entsprechend anpassen können. Das kann mit oder ohne das aktive Eingreifen des Nutzers geschehen. Passive Personalisierung kann beispielsweise in dem automatischen Ausblenden nie verwendeter Funktionen bestehen (vgl. Blair-Early/Zender 2008, 102).

IN-01: Aktive Personalisierung anbieten
Die Applikation soll dem Nutzer erlauben, Einstellungen an einer Funktion vorzunehmen.

IN-02: Passive Personalisierung anbieten
Die Applikation soll dem Nutzer nach einer bestimmten Zahl an Nutzungen von einer Funktion die Möglichkeit bieten, nicht verwendete Optionen auszublenden.

Neben der Komplexität und Nutzungshäufigkeit der Funktion spielt der Erfahrungsstand des Nutzers eine Rolle bei der Individualisierung. Wie in Kapitel 4.3 beschrieben, empfiehlt es sich, Anfängern eine Variante mit wenigen Optionen und Experten eine Variante mit vielen Optionen anzubieten. Unerfahrene Nutzer möchten bei der Aufgabenerfüllung angeleitet werden und sollten daher einen geführten, geradlinigen Funktionsablauf erhalten (zum Thema Hilfestellung siehe auch Kapitel 4.15 *Hilfe anbieten*). Erfahrene Nutzer hingegen würden durch diesen Ablauf verlangsamt werden. Sie wünschen sich stattdessen Abkürzungen, um schneller ihre Aufgaben zu erledigen. Diese Abkürzungen können in dem Überspringen von Teilschritten oder dem gleichzeitigen Ausführen mehrerer Teilschritte mit nur einem Befehl bestehen (vgl. Scapin/Bastien 1997, 226; Lin/Choong/Salvendy 1997, 271; Nielsen 1993, 41).

IN-03: Mehrere Aktionen gleichzeitig ausführen
Die Applikation soll das gleichzeitige Ausführen mehrerer Aktionen durch eine einzige Aktion ermöglichen.

Der richtige Grad an Individualisierbarkeit eines Systems lässt sich nicht pauschal an einigen wenigen Kenngrößen festmachen. Es ist hier, wie zuvor erwähnt, eine ausgedehnte Nutzerkenntnis notwendig. Mit Tests oder Beobachtungen lässt sich feststellen, ob Nutzer einen oder mehrere Wege zur Aufgabenerfüllung benötigen. Sind die Bedürfnisse der Nutzer eindeutig bekannt, lassen sich Funktionen genauestens auf diese Bedürfnisse spezialisieren. Wenn nur bekannt ist, dass eine Software stark unterschiedliche Nutzergruppen hat, empfiehlt sich das Anbieten verschiedener Navigationsformen (vgl. Lidwell/Holden/Butler 2003, 102; Stapelkamp 2010, 310). Hier können ebenfalls mit Tests oder dem systematischen Beobachten der realen Benutzung die bevorzugten Navigationsformen der Anwender gefunden werden (Lidwell/Holden/Butler 2003, 76).

4.4 Fehler-Handhabung

Ein von vielen Autoren genannter Usability-Bereich handelt von dem richtigen Umgang mit Fehlern. Wenngleich hohe Fehlertoleranz als Merkmal guter Usability gilt und viele der anderen genannten Bereiche hierzu beitragen, gibt es auch eine Reihe eigener Empfehlungen für das Schaffen einer guten Fehler-Handhabung.

Fehler sind im weitesten Sinne Aktionen, die ein unbeabsichtigtes Ergebnis hervorbringen (vgl. Lidwell/Holden/Butler 2003, 82). Es gilt allgemein die Empfehlung, ein Produkt so zu gestalten, dass nur wenige Fehler passieren. Falls doch Fehler geschehen, soll das System eine schnelle Erholung ermöglichen (vgl. Jordan 1998, 30; Nielsen 1993, 32f.). Grund dafür ist, dass Fehler den Arbeitsfluss des Nutzers stören, was ihn wiederum an der Ausführung seiner Aufgabe hindert. Eine hohe Fehlerzahl wirkt sich somit negativ auf Effektivität und Effizienz der Software aus (vgl. Scapi/Bastien 1997, 226). Auch die ISO-Norm 9241 beschäftigt sich mit diesem Thema. Sie definiert die sog. Fehlertoleranz wie folgt:

> „Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand durch den Benutzer erreicht werden kann.“ (International Standards Organization 2006)

Die Fehler, die hier betrachtet werden, werden durch Fehlverhalten des Nutzers ausgelöst. Fehler, die der mangelhaften Funktionalität des Systems zuzuordnen sind, werden hier nicht behandelt. Um Nutzungsfehler zu verstehen, müssen deren Ursachen verstanden werden. Sogenannte „Ausrutscher“ passieren, wenn Handlungen falsch ausgeführt werden. Es handelt sich hier entweder um Versehen oder um Handgriffe, die automatisch oder ohne große Konzentration durchgeführt werden und bei Unterbrechung der üblichen Routine misslingen. Hinter „Irrtümern“ steckt dagegen eine falsche Absicht, die zu einer unpassenden Handlung führt. Irrtümer entstehen durch Stress oder Voreingenommenheit (vgl. Lidwell/Holden/Butler 2003, 82). Das richtige Handhaben von Fehlern umfasst drei Bereiche: das Vermeiden von Fehlern im Vorhinein, die Qualität der Fehlerbenachrichtigungen und das Korrigieren von Fehlern (vgl. Scapin/Bastien 1997, 227).

Zunächst soll ein System so gestaltet werden, dass die Zahl möglicher Ausrutscher von vornherein minimiert wird (Jordan 1998, 30). Hier sind die Empfehlungen zu nennen, die in den anderen Kapiteln vorgestellt wurden: das Anbieten von Austritts- oder Abbruchmöglichkeiten, das Beschränken des Handlungsspielraumes (Kapitel 4.1), das Gestalten schlanker, kurzer Interaktionsabläufe (Kapitel 4.2) und das deutliche Anzeigen von Rückmeldungen auf Aktionen (Kapitel 4.10) sowie des aktuellen Systemstatus’ (Kapitel 4.9) tragen deutlich zu einer bewussten, Ausrutscher-freien Nutzung bei. Um Irrtümer und Missverständnisse des Nutzers zu vermeiden, sollte das Design klar, übersichtlich und nicht zu überladen sein. Eindeutige und verständliche Formulierungen helfen dabei, die richtigen Entscheidungen bei der Nutzung zu treffen (vgl. Lidwell/Holden/Butler 2003, 82). Näheres zu diesem Bereich findet sich in Kapitel 4.14.

Eine weitere Technik, um Fehler im Vorhinein zu vermeiden, ist das sog. „Bestätigen kritischer Aktionen“. Kritische Aktionen können z. B. das Löschen von Informationen oder das Abschließen eines Kaufvertrags sein. Ein weiteres Beispiel ist das Eingreifen in den Code einer Software, das irreparablen Schaden anrichten kann. Es handelt sich hier vorrangig um Schritte, die nur schwer oder gar nicht rückgängig gemacht werden können. Während bei normalen Funktionen des Systems das einmalige Ausführen einer Aktion ausreicht, sollen kritische Aktionen ein zweites Mal bestätigt werden. Dadurch fragt das System den Nutzer, ob die Aktion wirklich beabsichtigt ist und erhält die Möglichkeit, auf eventuelle Folgen der Aktion hinzuweisen. Es entsteht ein direkter Dialog mit dem Nutzer, der bei Software oft durch Dialogfelder mit der Frage

„Sind sie sicher, dass ...“ gestaltet wird. Auf diese Weise werden unbeabsichtigte Aktionen vermieden und die Fehlerwahrscheinlichkeit gesenkt (vgl. Lidwell/Holden/Butler 2003, 54; Ludewig/Lichter 2010, 371).

> **FE-01:Bestätigung kritischer Aktionen**
> Löst der Nutzer eine kritische Aktion aus, soll die Applikation einen Dialog öffnen, in dem der Nutzer die Aktion ein zweites Mal bestätigen oder abbrechen kann.

Der zweite Bereich der Fehlerhandhabung ist das Anbieten guter Fehlermeldungen. Geschieht ein Fehler, muss das System den Nutzer auf diesen hinweisen. Dabei ist – wie in Kapitel 4.14 beschrieben – auf die Sprache zu achten: der Hinweis sollte über die Aufgabe sprechen und nicht über Zustände im System; der Nutzer sollte die Formulierungen verstehen, was eine gute Kenntnis der Nutzer voraussetzt (vgl. Hix/Hartson 1993, 41). Es wird außerdem empfohlen, die Fehlermeldung so schnell wie möglich erscheinen zu lassen, also unmittelbar nach oder während der Falscheingabe. Es sind Dialoge möglich, in denen die Nutzer erst verschiedene Eingaben vornehmen und dann eine Aktion durchführen kann (z. B. das Ausfüllen von einem Formular). In dem nicht seltenen Fall, dass ein einzelnes Eingabefeld falsch ausgefüllt wurde, kann die Software den Fehlerhinweis entweder beim versuchten Ausführen der Funktion anzeigen oder schon während der Eingabe. Letztere Variante ist bequemer für den Nutzer, weil er seine Falscheingabe korrigieren kann, ohne im Dialog zurückgehen zu müssen (vgl. Jordan 1998, 30). Darüber hinaus sollten die Fehlerhinweise spezifisch und konstruktiv sein. Harte, negative Formulierungen verängstigen oder frustrieren den Nutzer; die Fehlermeldung sollte den Nutzer unterstützen und nicht bestrafen. Dabei sollten auch humorvolle Formulierungen vermieden werden, da Nutzer aufgrund ihres Fehlers vermutlich negativ gestimmt sind (vgl. Hix/Hartson 1993, 41f.). Um die vom Anwender wahrgenommene Kontrolle aufrecht zu erhalten, sollte das System bei der Formulierung von Fehlermeldungen die Schuld auf sich nehmen, wann immer dies möglich ist. Beispielsweise kann eine falsche Befehlseingabe mit „Unzulässiger Befehl“ oder „Unbekannter Befehl“ beschrieben werden. Ersteres gibt dem Nutzer das Gefühl, etwas Falsches getan zu haben, während letzteres dem System die Schuld gibt (vgl. Hix/Hartson 1993, 43). Eine konstruktive Fehlermeldung, die den Nutzer zum gewünschten Verhalten ermutigt, trägt unmittelbar zur Lernbarkeit des Systems bei (Scapin/Bastien 1997, 227).

FE-02: Fehler-Benachrichtigungen
Macht der Nutzer einen Fehler, soll die Applikation ihn unmittelbar darauf hinweisen.

Zeigt die Applikation eine Fehler-Benachrichtigung an, soll diese explizite Hinweise zur Problemlösung enthalten.

Der letzte Bereich der Fehlerhandhabung ist das schnelle Erholen nach geschehenen Fehlern. Hierzu gehören die in Kap 4.1 genannten Steuerungsmöglichkeiten. Wenn ein Nutzer sich in einer Schrittfolge bewegt, die er jederzeit abbrechen kann, wird er Fehler leicht korrigieren können. Zusätzlich sollte der Nutzer die Möglichkeit haben, Aktionen und Eingaben direkt rückgängig zu machen, ohne den aktuellen Dialog abbrechen zu müssen. In Kombination mit den hilfreichen Fehlermeldungen kann er so mühelos Fehler korrigieren und dabei im Arbeitsfluss bleiben. Voraussetzung ist, dass die Korrekturmöglichkeit genauso unmittelbar verfügbar ist wie die Fehlermeldung (Reeves et al. 2004, 59; Scapin/Bastien 1997, 227). Das Prinzip, alle Eingaben grundsätzlich rückgängig machen zu können, wird von manchen Autoren als „Forgiveness" bezeichnet. Forgiveness erlaubt Nutzern, die Bedienoberfläche gefahrlos auszuprobieren, was ebenfalls zur Lernbarkeit des Systems beiträgt (vgl. Blair-Early/Zender 2008, 99; Jordan 1998, 30; Hix/Hartson 1993, 45).

FE-03: Aktionen rückgängig machen
Der Nutzer soll die Möglichkeit haben, Aktionen unmittelbar rückgängig zu machen.

Sind in einer Funktion mehrere Eingaben durch den Nutzer zu tätigen, sollten im Falle eines Fehlers nur die falschen Eingaben ein zweites Mal bearbeitet werden. Dies trägt zu Effizient und Bequemlichkeit der Nutzung bei (vgl. Scapin/Bastien 1997, 227).

FE-04: Isoliertes Korrigieren von falschen Eingaben
Macht der Nutzer Fehler, soll er die falschen Eingaben unmittelbar korrigieren können, ohne die richtigen Eingaben neu eingeben zu müssen.

An Funktionen mit Eingabeformularen kann zusätzlich die detailliertere Anforderung gestellt werden, dass alle Eingaben vor Abschicken des Formulars verändert werden können (vgl. Scapin/Bastien 1997, 227).

FE-05: Überprüfen von Eingaben vor Auslösen der Funktion
Der Nutzer soll die Möglichkeit haben, alle Eingaben eines Eingabeformulars zu überprüfen, bevor er die dazugehörige Funktion auslöst.

Um Fehler korrigieren zu können, sollten dem Nutzer einmal getätigte Eingaben, die er zwischenzeitlich verändert oder gelöscht hat, zur Verfügung stehen (vgl. Connel et al. 1997, 35). Dazu sollte das System eingegebene Inhalte speichern. Dies sollte idealerweise so ablaufen, dass der Nutzer nicht aktiv Speichervorgänge auslösen muss, sondern das System selbstständig Informationen aufbewahrt, ohne dass der Nutzer dies merkt. Lediglich der Aufbewahrungsort der Informationen sollte dem Nutzer bekannt sein (vgl. Knittle/Ruth/Gardner 1987, 171).

FE-06: Automatisches Speichern überschriebener Inhalte
Wenn der Nutzer einmal erstellte Inhalte ändert, soll die Applikation die vorherigen Inhalte ohne Aufforderung des Nutzers speichern und abrufbar machen.

4.5 Eintrittspunkte

Ein Eintrittspunkt oder Entry Point ist der Punkt, an dem ein Nutzer in ein Design eintritt. Bei Software ist das üblicherweise die Anzeige, die erscheint, nachdem das Programm vollständig geladen wurde und zur Nutzung bereit ist. Bei Webseiten ist es die Startseite, die bei Eingabe der Internetadresse erscheint. Eintrittspunkte sind wichtig für Usability, weil hier die Nutzung des Systems beginnt und weil für Menschen der erste Eindruck von einer Sache eine große Rolle spielt. Eintrittspunkte beeinflussen die Wahrnehmung des Nutzers für alle nachfolgenden Schritte, sofern es die gibt (vgl. Lidwell/Holden/Butler 2003, 80; Kaasinen 2005, 89). Es ist daher ratsam, eine Reihe von Anforderungen bei der Gestaltung eines Eintrittspunktes zu beachten.

Die erste Empfehlung lautet, überhaupt einen Eintritts- oder Ausgangspunkt für die Interaktion anzubieten. Wenn es einen offensichtlichen Startpunkt gibt, kann der Nutzer lernen, sich in dem Design zurechtzufinden. Dadurch wird das Design fehlertoleranter und vermittelt dem Nutzer Sicherheit. Ein Beispiel sind die Startseiten von Internetseiten, die jederzeit über einen Klick auf das Logo der Seite im oberen Seitenbereich erreicht werden können (vgl. Blair-Early/Zender 2008, 99).

EP-01: Startpunkt anbieten
Die Navigation soll einen offensichtlichen Startpunkt haben.

EP-02: Jederzeitiges Zurückkehren zum Startpunkt
Der Nutzer soll jederzeit die Möglichkeit haben, zum Startpunkt zurückzukehren.

Eintrittspunkte sollten minimale Barrieren aufweisen. Eine Barriere kann in einer hinderlichen Funktionalität bestehen oder in ästhetischen Elementen, die abschreckend wirken. Gut benutzbare Designs laden ihre Nutzer zum Benutzen ein und bremsen sie nicht mit wahrgenommenen Hindernissen (vgl. Lidwell/Holden/Butler 2003, 80). Manche Programme oder Webseiten arbeiten mit Startbildschirmen, sog. „Splash Screens", die der eigentlichen Startseite mit der Hauptnavigation vorgeschaltet sind. In der Vergangenheit wurden solche Startbildschirme genutzt, um mit Hilfe von Animationen und/oder Geräuschen bei Benutzern eine gewisse Stimmung auszulösen und den Fokus auf eine Willkommensnachricht oder die Marke zu lenken. Splash Screens werden kritisiert, weil sie den Arbeitsfluss des Nutzers verlangsamen können und damit der Forderung nach Effizienz entgegen stehen. Im Internet versuchen Nutzer oft, diese Bildschirme so schnell wie möglich zu überspringen. Da diese Startbildschirme eine Barriere darstellen, sollten sie nur eingesetzt werden, wenn eine Barriere gebraucht wird, z.B. bei Webseiten mit vertraulichem oder nicht-jugendfreiem Inhalt (vgl. Cappel/Huang 2007, 118f.; Nielsen 2000, 176).

EP-03: Keine Barrieren im Startpunkt
Der Startpunkt der Applikation soll das unmittelbare Verwenden ihrer Funktionen erlauben.

Der Startpunkt einer Nutzeroberfläche sollte dem Nutzer einen klaren Überblick über die zu erreichenden Informationen oder Funktionen geben. Es sollten nicht zu viele Informationen angezeigt werden, die nicht zum eigentlichen Service gehören. Das Schaffen eines klaren Überblicks über die Bereiche der Software wurde mit dem Aufkommen von internetfähigen Handys erneut zur Herausforderung, da klassische Internetangebote auf kleinen Bildschirmen einen schlechten Überblick boten. Das Neufassen von Hauptmenüs und Navigationen ist seither eine wichtiger Arbeitsgegenstand der Mobiloptimierung von Webservices (vgl. Kaasinen 2005, 89; Alby 2008, 118ff.; Lidwell/Holden/Butler 2003, 80).

EP-04: Überblick-Funktion des Startpunkts

Der Startpunkt der Navigation soll einen Überblick über die Funktionen der Applikation enthalten.

Schlussendlich sollte der Eintrittspunkt den Nutzer dazu bringen, das System zu nutzen, sich also von der Startseite aus tiefer in das Design hinein zu bewegen. Dabei soll mit sog. „Verlockungen" gearbeitet werden, die auf ansprechende Weise die Aufmerksamkeit des Nutzers auf sich ziehen und ihn durch das gesamte Design führen (vgl. Lidwell/Holden/Butler 2003, 80). Da bei Lockmitteln vor allem mit ästhetischen und sprachlichen Elementen gearbeitet wird, lässt sich hierfür keine überprüfbare Anforderung formulieren.

4.6 Wegfindung

Wie zuvor beschrieben zeigt gut benutzbare Software explizit, wozu sie zu welchem Zeitpunkt genutzt werden kann und welche Funktionen aktuell genutzt werden können (vgl. Jordan 1998, 37). Dieser Anforderungsbereich wird von einigen Autoren mit Empfehlungen zum Thema Wegfindung ergänzt.

Wegfindung ist der Prozess, bei dem Umgebungsinformationen genutzt werden, um sich zu orientieren und um zu einem Ziel zu gelangen. Unabhängig davon, ob Wegfindung in der Realität oder in einer Software stattfindet, werden vier Prozesse durchlaufen: Orientierung, Entscheidung für einen Weg, Kontrollieren des Wegs und Erkennen des Ziels (vgl. Lidwell/Holden/Butler 2003, 260). Eine Software mit guter Usability unterstützt ihre Anwender bei jedem dieser Prozesse. Um dies zu erreichen, müssen verschiedene Empfehlungen beachtet werden, die teilweise schon in vorangegangenen Kapiteln genannt wurden. Die ISO-Norm 9241 beschreibt Selbstbeschreibungsfähigkeit als ein Attribut, das zur Orientierung und bewussten Nutzung einer Webseite beiträgt (International Standards Organization 2006):

> „Ein Dialog ist selbstbeschreibungsfähig, wenn jeder einzelne Dialogschritt durch Rückmeldung des Dialogsystems unmittelbar verständlich ist oder dem Benutzer auf Anfrage erklärt wird."

In dieser Definition werden die Darstellung des Systemstatus' und die unmittelbare, aussagekräftige Rückmeldung des Systems genannt; beide Punkte werden in späteren Kapiteln ausgeführt. Es wird außerdem darauf hingewiesen, dass Nutzer wissen

müssen, wo sie herkommen, wo sie sind und wohin sie navigieren können. Um den Anwender bei der Orientierung zu unterstützen, sollte die Software Orientierungspunkte anbieten, durch die er erkennt, wo und auf welcher Ebene der Navigation er sich befindet (Rampl 2007). Einige dieser Orientierungspunkte sollten zu jedem Zeitpunkt sichtbar sein. Sie tragen dazu bei, dass der Anwender den Aufbau der Software in seinem Kopf nachbauen kann, also ein mentales Modell errichtet und die Nutzung besser lernt (vgl. Blair-Early/Zender 2008, 101). Bei klassischen Webseiten oder Dateiverwaltungsprogrammen wie beispielsweise dem Windows-Explorer wird dies über sog. „Brutkrumen-Spuren" oder Sitemaps gelöst. Dabei wird der gegangene Pfad des Nutzers mit jeder genutzten Hierarchieebene angezeigt (vgl. Preece 2000, 278, Cappel/Huang 2007, 119). Eine einfachere Variante wäre, einfach nur jeder aufrufbaren Seite einen Titel zu geben (vgl. Scapin/Bastien 1997, 222).

WE-01: Orientierung anbieten
Die Applikation soll dem Nutzer zu jedem Zeitpunkt anzeigen, an welcher Stelle der Navigation er sich befindet.

Darüber hinaus sollte der Nutzer erkennen können, wohin ihn die angebotenen Navigationselemente bei deren Benutzung führen. Die Benennung der Navigationselemente sollte deshalb aussagekräftig sein (Rampl 2007).

WE-02: Aussagekräftige Bedienelemente
Die Navigation der Applikation soll aussagekräftig und klar benannte Bedienelemente haben.

Hat der Nutzer sich für einen Weg entschieden, wird er eine Aktion ausführen. Hier kommen die in anderen Kapiteln genannten Hilfen zum Tragen: das System gibt ihm direkt nach Ausführen der Aktion ein aussagekräftiges Feedback, anhand der klaren Menüführung, der geordneten Informationsdarstellung und dem angezeigten Systemstatus weiß er, welche Optionen ihm offen stehen.

Es kann vorkommen, dass ein Anwender während der Nutzung die Orientierung verliert oder Fehler macht, die er nicht zu korrigieren weiß. Neben den bereits genannten Abbruchsmöglichkeiten und den Hilfefunktionen kann eine Software eine Suchleiste einsetzen, um orientierungslosen Nutzern eine Hilfe anzubieten (vgl. Cappel/Huang 207, 119). Dies ist jedoch eine Lösungsvariante, deren Angemessenheit von der jeweiligen Software abhängt.

4.7 Anordnung von Informationen

Die Anordnung von Informationen auf einer Bedienoberfläche ist in vielerlei Hinsicht wichtig für eine gute Usability. Als Informationen werden in diesem Kapitel sowohl Inhalte als auch Bedienelemente verstanden, die aus Text und Bildern bestehen können. Dies schließt die Menüführung in Bedienoberflächen ein. Zunächst wird an die in Kapitel 2.2 genannten Usability-Ziele erinnert: Anwender sollen die zu entwickelnde Software effizient nutzen können und dabei die geringstmögliche kognitive Belastung erleben. Um dies zu erreichen, soll eine Bedienoberfläche den Fokus der Aufmerksamkeit stets auf wichtige Informationen und Elemente lenken und weniger wichtige Informationen verstecken (Ludewig/Lichter 2010, 371; Connel et al. 1998, 34). Es wird grundsätzlich empfohlen, Informationen ihrer Wichtigkeit entsprechend anzuordnen, um eine gute Usability zu erreichen (vgl. Lidwell/Holden/Butler 2003, 140).

In einem ersten Schritt muss herausgefunden werden, wie wichtig die einzelnen Informationen oder Elemente der jeweiligen Applikation sind. Die Wichtigkeit von Informationen und Elementen kann sich daraus ergeben, wie essentiell sie für die Nutzung der Applikation sind oder wie oft sie verwendet werden (vgl. Jordan 1998, 34). In einem zweiten Schritt sollte eine Hierarchie der Informationen festgelegt werden, die im ganzen System durchgehalten wird. Hierarchien haben den Vorteil, dass sie komplexe Informationsstrukturen auf sehr einfache Weise für Anwender verständlich machen. Typische Visualisierungsformen sind Baum- oder Treppenstrukturen, z. B. in Form von Ordnern und Unterordnern bei einem Computer-Betriebssystem (vgl. Lidwell/Holden/Butler 2003, 122). An vorderster Stelle einer Bedienoberfläche sollten stets die Kernelemente stehen. Dieses Herausstellen der Schlüsselinformationen hat eine Reihe von Vorteilen: die Kernaspekte der Software sind klar erkennbar und werden leichter erinnert, das Zurechtfinden im System wird erleichtert und eine schnelle Nutzung wird ermöglicht (vgl. Lidwell/Holden/Butler 2003, 140). Dies ist v.a. bei dem in Kapitel 4.5 beschriebenen Entwickeln von Eintrittspunkten entscheidend, wo Nutzern die Hauptfunktionen der Software schnell klar werden sollen. Neben dem aktiven Betonen wichtiger Elemente können auch die weniger wichtigen in den Hintergrund gestellt werden. Ob ein verkleinertes Darstellen dieser Elemente oder ein komplettes Ausblenden oder Verstecken sinnvoll ist, ist Thema der Problemlösung. Durch zurückgestellte Darstellung unwichtiger Informationen gewinnt die Benutzeroberfläche an Übersichtlichkeit und wirkt weniger

überladen, kann also vom Betrachter schneller erfasst werden. Das Herausstellen wichtiger Informationen kann so weit betrieben werden, dass die Hauptfunktionen der Software jederzeit sichtbar und aufrufbar sind (vgl. Hix/Hartson 1993, 49; Jordan 1998, 43; Scapin/Bastien 1997, 224f.).

AI-01: Wichtige Informationen hervorheben
Die Applikation soll wichtige Informationen gegenüber weniger wichtigen Informationen hervorgehoben darstellen.

Neben der Hierarchisierung können komplexe Informationen und Elemente gruppiert werden, um dem Nutzer eine Struktur anzubieten. Dies kann z. B. mit räumlicher Nähe und gleichen Farben bei der Darstellung geschehen. Je nachdem, wie nah und ähnlich oder fern und unähnlich Objekte dargestellt werden, wird ihre Beziehung vom Nutzer als zusammengehörig empfunden. Das Gleiche gilt für Objekte, die sich in der gleichen Weise bewegen. Werden Objekte mit Hilfe solcher Darstellungsformen geordnet, erkennt ein Nutzer dies als Gruppierung und kann die Benutzeroberfläche schneller erlernen (Scapin/Bastien 1997, 222f.; Blair-Early/Zender 2008, 101f.). Es sei angemerkt, dass Gruppierungen sich v.a. bei vielfältigen oder komplexen Informationen anbieten (vgl. Lidwell/Holden/Butler 2003, 146).

AI-02: Informationen gruppieren
Die Applikation soll Informationen in der Darstellung gruppieren.

Das Ziel der Anwendung solcher Strukturierungsmethoden ist auch, die entstehenden Menüs so simpel wie möglich zu halten. Dies ist v. a. für mobile Applikationen entscheidend: je einfacher die Menüstruktur, desto effektiver und fehlerfreier navigieren Nutzer durch die Applikationen (vgl. Ziefle 2002, 310). Das bloße Nutzen von Hierarchien und Gruppierungen in Menüs ist noch kein Garant für einfache Bedienoberflächen. Hierarchie-Ebenen erzeugen Tiefe. Der Vorteil hierbei ist, dass Informationen so ihrem Detailgrad entsprechend sortiert werden können; der Nachteil ist eine größere Fehlerwahrscheinlichkeit und ein häufigeres Hin- und Herbewegen. Gruppierungen erzeugen hingegen Menübreite. Hierbei geschehen weniger Navigationsfehler, jedoch können viele Gruppierungen ein Menü überladen (Kim/Jacko/Salvendy 2011, 390). Es gibt unterschiedliche Meinungen darüber, welche Form der Menüstrukturierung sich besser für bestimmte Endgeräte eignet. Während bei Computern viele Elemente auf einmal angezeigt werden können, haben Smartphone-Bildschirme nur eine geringe Darstellungsfläche. Hier wird empfohlen,

sowohl Tiefe als auch Breite der Menüs zu reduzieren und dafür mehr Menü-Elemente auf einem Bildschirm anzubieten, damit Scrollen verhindert wird (vgl. Huang 2006, 106; Kim/Jacko/Salvendy 2011, 398f.). Alternativen bestehen auch bei der flächenmäßigen Größe des gezeigten Inhalts wie der Länge der Seiten im (mobilen) Internet: einerseits sind kurze Seiten möglich, die durch viel Navigation miteinander verbunden sind und wenig Scrollen erfordern. Andererseits können längere Seiten mit weniger Menüpunkten und mehr Scrollbedarf erstellt werden. Scrollen wird von vielen Anwendern als angenehmer empfunden als das Wechseln einer Seite. Dies gilt v.a. bei langsamen Anwendungen, schlechter Internetqualität und großen Bildschirmen. Bei kleinen Bildschirmen und schneller Datenverarbeitung werden kürzere Seiten empfohlen. Scrollen ist unbeliebt, wenn Nutzer einen bestimmten Inhalt suchen. Haben sie ihn jedoch gefunden, scrollen sie gerne, um ihn ganz zu betrachten oder zu lesen. Es gilt grundsätzlich die Regel, die entscheidenden Informationen mehr herauszustellen, je kleiner der verwendete Bildschirm ist (vgl. Kaasinen 2005, 90f.). Unabhängig vom Endgerät wird empfohlen, die Navigation der Software stets erreichbar zu machen, ohne dass diese durch Scrollen sichtbar gemacht werden muss (vgl. Huang 2006, 106; Kim/Jacko/Salvendi 2011, 398f.).

AI-03: Ständig erreichbare Navigation
Die Navigation der Applikation soll immer erreichbar sein.

4.8 Konsistenz

Eine hohe Konsistenz der Elemente eines Systems ist einer der am häufigsten genannten Faktoren guter Usability. Konsistenz bedeutet, dass ähnliche Teile eines Systems auf ähnliche Weise dargestellt werden und ähnlich funktionieren. Dies erleichtert es Nutzern, das System zu lernen, da neue, unbekannte Systemkomponenten schneller erfasst werden können und die Aufmerksamkeit schneller auf die wichtigen Inhalte oder die aktuelle Aufgabe gelenkt werden kann (Lidwell/Holden/Butler 2003, 56; Scapin/Bastien 1997, 227; Rampl 2007). Es kann in drei Arten der Konsistenz unterschieden werden (vgl. Lin/Choong/Salvendy 1997, 270f.; Lidwell/Holden/Butler 2003, 56):

- **Funktionale Konsistenz** bezieht sich auf die Konsistenz von Bedeutung und Handlung, wie z.B. bei den Tasten eines Videorekorders: die Symbole für Abspielen oder Pause bedeuten bei allen möglichen Abspielmedien das

Gleiche. Durch die konsistente Symbolik können Nutzer das Wissen eines bestimmten Bereiches auf einen anderen Bereich übertragen.

- **Innere Konsistenz** ist das Durchhalten von Designentscheidungen bei allen Komponenten eines Systems, z.B. in Form gleicher Symbolik, gleicher Benennungen etc.
- **Externe Konsistenz** ist das Durchhalten von Designentscheidungen über verschiedene Systeme oder Plattformen hinweg.

Ein konsistentes Design hat Wiedererkennungswert und wird als vollwertiges Produkt wahrgenommen, während Designbrüche beim Nutzer den Eindruck eines hastig zusammengebauten Systems erwecken (vgl. Lidwell/Holden/Butler 2003, 56). Die immer gleichen Gestaltungselemente machen das System für den Nutzer berechenbar und damit besser lernbar. Gleichzeitig verringert sich die Zahl an Fehlern während der Nutzung, da Nutzer neue Bereiche des Systems besser einzuordnen wissen (Scapin/Bastien 1997, 227; Jordan 1998, 25f.; Blair-Early/Zender 2008, 100). Bei den genannten Designentscheidungen sind alle denkbaren Gestaltungsfelder möglich, von Benennungen und Sprache, visueller Gestaltung, Befehlen, Reihenfolgen bis hin zu Formaten und Dateneingabefeldern (vgl. Scapin/Bastien 1997, 227; Ludewig/Lichter 2010, 371; Rosson/Carrol 2002, 155). Dementsprechend kann eine ganze Reihe Empfehlungen formuliert werden, die einen unterschiedlich hohen Detailgrad aufweisen. Möglich sind Standards der Sprache, der Farbe oder der Schriftwahl (vgl. Nassar 2012, 1055).

Grundsätzlich sollen alle Bereiche einer Applikation eine ähnliche Gestaltung haben. Wechselt ein Nutzer von einem Bereich der Anwendung zum nächsten, soll er diesen Wechsel bestenfalls gar nicht wahrnehmen und seine Aufgabe ohne Unterbrechung weiter verfolgen können. Ist dies nicht möglich, weil ein System beispielsweise ein eingebautes, fremdes Modul enthält, sollte der Nutzer auf diesen Umstand hingewiesen werden. Ein fremdes Modul kann beispielsweise ein Plug-in in einem Internetbrowser oder ein Suchmaschinenfeld auf einer Internetseite sein (vgl. Preece 2000, 280). Um Anforderungen für konsistente Gestaltung überprüfbar zu machen, wird die ähnliche Gestaltung mit „identisch formatiert“ umschrieben.

CO-01: Alle Teile einer Applikation konsistent gestalten
Teilt sich die Applikation in verschiedene Bereiche auf, sollen die Bereiche identisch formatiert sein.

Es können Konsistenz-Anforderungen für die Funktionen des Systems verwendet werden. Gleiche oder ähnliche Operationen sollen grundsätzlich konsistent gestaltet werden (vgl. Ludewig/Lichter 2010, 371):

> **CO-02: Konsistent gestaltete Funktionen**
> Bietet die Applikation an unterschiedlichen Positionen gleiche oder ähnliche Funktionen an, sollen die Funktionen überall identisch formatiert sein.

Ähnliche Anforderungen lassen sich für die konsistente Darstellung von Informationen und anderen Elementen der Bedienoberfläche formulieren (vgl. Ludewig/Lichter 2010, 371; Reeves et al. 2004, 58):

CO-03: Konsistent gestaltete Informationen

Bietet die Applikation an unterschiedlichen Positionen gleiche oder ähnliche Informationen an, sollen die Informationen überall identisch formatiert sein.

Externe Konsistenz kommt bei Software v.a. dann zum Tragen, wenn eine Software mit unterschiedlichen Endgeräten genutzt wird. Das ist grundsätzlich bei allen Internetanwendungen der Fall, allerdings auch bei zu installierenden Computer-Programmen oder Smartphone-Applikationen. Schon früh wurde gefordert, Bedienoberflächen auf jeder Bildschirmgröße möglichst gleich anzuzeigen (vgl. Hix/Hartson 1993, 48). Diese Forderung ist heute aktueller denn je, da mit Tablett-PCs und Smartphones unterschiedlichste Bildschirmgrößen und Bedienungsformen im Umlauf sind. Seither sind neue Arbeitsfelder in der Mediengestaltung wie beispielsweise Mobiloptimierung von Webseiten oder Responives Design aufgetaucht, die sich mit der Anpassung von klassischen Web-Inhalten an neue Bildschirmformate beschäftigen (Alby 2008, 103ff.; Kaasinen 2005, 100). Es lässt sich folgende standardisierte Anforderung ableiten:

> **CO-04: Konsistente Darstellung auf verschiedenen Endgeräten**
> Wird eine Applikation von unterschiedlichen Endgeräten aus genutzt, soll die Applikation auf jedem Endgerät identisch dargestellt werden.

Die Forderung nach hoher innerer und externer Konsistenz ist keine Forderung nach gestalterischer Monotonie: solange die Logik des Inhalts, der Aktionen und deren Effekte immer ähnlich ist, bleiben durchaus gestalterische Spielräume. In einem System kann sich beispielsweise an einigen Stellen der Inhalt verändern, indem er

umfangreicher und detaillierter wird. In diesem Fall müsste sich auch das Interface anpassen, da es dem größeren Detailgrad gerecht werden müsste. Bei einer solchen Veränderung der Bedienoberfläche müsste allerdings nicht mit der Logik der sonstigen Systembedienung gebrochen werden (vgl. Blair-Early/Zender 2008, 100).

Die wichtigste Form der Konsistenz ist die Übereinstimmung des Systems mit den Erwartungen der Nutzer (vgl. Blair-Early/Zender 2008, 100). Konsistenz kann daher über eine bestimmte Software hinausreichen. Wenn ein in sich konsistentes Design von einem Nutzer gelernt wird, entstehen bei ihm Erwartungen bezüglich der Nutzung. Solche Erwartungen können jedoch auch von fremden Systemen oder Umgebungen kommen. Die in diesem Rahmen zu nennenden Empfehlungen werden in Kapitel 4.11 diskutiert.

4.9 Sichtbarkeit des Systemstatus

Um die Benutzbarkeit eines Systems zu erhöhen, sollte der aktuelle Status des Systems angezeigt werden (vgl. Nielsen 1994, 30). Der Grund dafür ist, dass Anwender bei bekanntem Systemstatus wissen, welche Aktionen sie mit dem System aktuell ausführen können und welche Vorgänge zurzeit im System ablaufen. Menschen können Probleme besser lösen, wenn sie aus einer Reihe von Optionen wählen können, anstatt sich an bestimmte Lösungswege zu erinnern. Dementsprechend sollte eine Software immer die aktuell verfügbaren Optionen als solche zeigen. Bei komplexen Applikationen stehen oft unzählige kleine und große Funktionen zur Verfügung, mit mindestens ebenso vielen möglichen Ergebnissen. Dadurch wird die Zahl der möglichen Zustände, die im Rahmen des Systemstatus angezeigt werden könnten, sehr groß. Wenn versucht wird, das Anzeigen des Systemstatus’ in der Bedienoberfläche einzubringen, sollten nicht alle denkbaren Systemzustände angezeigt werden. Dies würde zu einer Informationsüberlastung führen, die der Benutzbarkeit in keiner Weise dienlich wäre. Stattdessen sollte versucht werden, die wichtigsten Vorgänge darzustellen. Das sind jene Vorgänge, die direkt mit der aktuellen Aufgabe zusammenhängen. Eine sinnvolle Eingrenzung ist es, nur Statusinformationen der aktuellen Hierarchie-Ebene anzuzeigen (zu Hierarchie-Ebenen siehe auch Kap. 4.7). Dadurch werden auf logische Weise über- und untergeordnete Zustände ausgeblendet und der aktuelle Nutzungskontext wird erfassbar (vgl. Reeves et al. 2004, 58f.; Lidwell/Holden/Butler 2003, 250f.; Nassar 2012, 1055f.).

SI-01: Systemstatus anzeigen
Die Applikation soll den Nutzer über den aktuellen Status des aktuell ausgewählten Bereiches informieren.

Ein prominentes Beispiel aus der Darstellung von Internetseiten ist die Farbe von Hyperlinks: auf klassischen Webseiten werden Links in blauer Schrift angezeigt; sobald sie einmal angeklickt wurden, wechselt ihre Farbe zu violett. Dies zeigt dem Nutzer auf einfache Weise, welche Bereiche er bereits besucht hat und welche ungenutzten Optionen ihm noch offen stehen (vgl. Cappel/Huang 2007, 119).

4.10 Rückmeldung

Eng verbunden mit dem Anzeigen des Systemstatus aus Kapitel 4.9 ist die Empfehlung, dem Nutzer bei allen von ihm durchgeführten Aktionen Rückmeldung zu geben. Dadurch wird das System für den Nutzer verständlicher, besser lernbar und gewinnt an Usability (vgl. Connel et al. 1998, 34).

Wenn ein Nutzer eine Aktion in der Software durchführt, also in irgendeiner Form einen Vorgang auslöst, sollte die Software unmittelbar eine wahrnehmbare Rückmeldung anzeigen (Palanque/Farance/Bastide 1999, 411). Wie stark diese Rückmeldung gestaltet wird, sollte mit der Wichtigkeit der Aktion zusammenhängen. Als Beispiel kann man Mauszeiger, die als Sanduhr das Arbeiten des Programms anzeigen, mit einer großen Einblendung nach Bestätigen eines Kaufvertrages vergleichen: beide Gestaltungsformen zeigen eine Rückmeldung an, unterscheiden sich jedoch in Größe und Bedeutung (vgl. Blair-Early/Zender 2008, 101). Dass die Rückmeldung unmittelbar erfolgt, also direkt nach Auslösen der Aktion, ist wichtig für die Gewissheit des Nutzers, dass seine Handlung einen Effekt hatte. Wenn nach der Eingabe zu lange nichts passiert, könnte der Nutzer den Eindruck gewinnen, dass das System nicht arbeitet. Unter Umständen löst er die Aktion ein zweites oder drittes Mal aus und erzeugt Fehler (vgl. Preece 2000, 280; Jordan 1998, 29; Blair-Early/Zender 2008, 101). Aus technischen Gründen ist es für Software oft nicht möglich, eine Aktion unmittelbar auszuführen. Wenn eine Aktion ausgelöst, aber noch nicht erfolgreich ausgeführt wurde, sollte dennoch eine direkte Rückmeldung über die begonnene Bearbeitung erfolgen. Das Vorliegen einer solchen Rückmeldung ersetzt dabei nicht das Schaffen einer schnell arbeitenden Software oder Web-Applikation; dauert das Ausführen einer Aktion zu lange, kann dies die Geduld des Nutzers auch mit angezeigter Rückmeldung ausreizen (vgl. Preece 2000, 280; Knittle/Ruth/Gardner

1987, 166). Wenn ein Vorgang sehr lange dauert, kann die Rückmeldung die Form einer Statusanzeige haben, durch die der Nutzer den Fortschritt verfolgen kann (vgl. Palanque/Farance/Bastide 1999, 411; Scapin/Bastien 1997, 223).

RM-01: Rückmeldung bei Aktionen
Wenn der Nutzer eine Aktion durchführt, soll die Applikation ihn unmittelbar und wahrnehmbar über die ausgelöste Bearbeitung informieren.

RM-02: Hinweis bei langer Bearbeitung
Wenn der Nutzer eine Funktion auslöst, deren Bearbeitung länger als 6 Sekunden dauert, soll die Applikation ihn unmittelbar und wahrnehmbar über die Bearbeitung informieren.

Wenn der Nutzer Eingaben in Form von Text, Ziffern, Tönen oder Bewegungen (z. B. bei einem Zeichenprogramm) macht, sollte die Software unmittelbar zeigen, dass diese Eingaben erfasst wurden. Bei Eingabefeldern können direkt die Eingaben angezeigt werden, es sei denn, es sind Felder zur Passworteingabe (vgl. Scapin/Bastien 1997, 223).

RM-03: Sofortiges Anzeigen von Eingaben
Wenn der Nutzer Eingaben in einem Eingabefeld macht, soll die Applikation unmittelbar anzeigen, ob und wie diese Eingaben im entsprechenden Eingabefeld erfasst wurden.

Schlussendlich sollte die Software den Nutzer nach abgeschlossener Bearbeitung darüber informieren, ob die Aktion erfolgreich ausgeführt wurde oder nicht. Beim Gestalten der Rückmeldung sollte darauf geachtet werden, dass die Rückmeldung den Nutzer über das informiert, was gerade im System geschieht. Das Freizeichen eines Telefons ist beispielsweise ein wenig aussagekräftiger Signalton, währenddessen die Tonabfolge beim Wählen der Tasten dem Nutzer erklärt, dass das Telefon die Zifferneingabe verarbeitet (vgl. Jordan 1998, 29). Diese Information über den Erfolg schließt auch den Fall des Misserfolgs oder des systemseitigen Abbruchs der Bearbeitung mit ein (vgl. Scapin/Bastien 1997, 23).

RM-04: Erfolgreiche Bearbeitung anzeigen
Wenn der Nutzer eine Funktion ausgelöst hat, soll die Applikation ihn nach der Bearbeitung unmittelbar und wahrnehmbar darüber informieren, ob die Funktion erfolgreich ausgeführt wurde oder nicht.

Durch das direkte Anzeigen solcher Rückmeldungen wird die Benutzung für den Anwender nachvollziehbarer und zufriedenstellender, da er das Gefühl hat, dass seine Handlungen etwas auslösen. Die daraus entstehende Sicherheit ist wichtig für die Vertrauensbildung zur Software (vgl. Hix/Hartson 1993, 39f.; Rampl 2007).

4.11 Erwartungskonformität

Die folgenden zwei Kapitel beschäftigen sich mit Gestaltungstechniken, die Erfahrungen und Vorstellungen der Anwender aufgreifen. Als Grundlage dafür soll zunächst der Begriff der mentalen Modelle erklärt werden.

Mentale Modelle sind geistige Vorstellungen des Menschen von der realen Welt, die auf gemachten Erfahrungen beruhen. Wenn Menschen ein System benutzen, gleichen sie unbewusst das reale Ergebnis dieser Benutzung mit dem zugehörigen mentalen Modell ab. Wenn die Ergebnisse gleich sind, ist das mentale Modell vollständig und genau. Menschen verstehen Systeme leichter, die mit ihren mentalen Repräsentationen übereinstimmen. Daraus leitet sich die Empfehlung für Gestalter ab, Systeme ähnlich diesen mentalen Modellen zu schaffen (vgl. Stapelkamp 2010, 59; Lidwell/Holden/Butler 2003, 154). Das Prinzip hinter dieser Empfehlung ist einfach: Anwendern fällt die Nutzung von Systemen leichter, von denen sie einige Teile bereits kennen, selbst wenn sie diese noch nie benutzt haben. Das Aufgreifen mentaler Modelle verringert die kognitive Belastung, beschleunigt das Erledigen der Aufgaben und trägt somit zu größerer Usability bei. Außerdem lernen Anwender schneller, je weniger sie neu lernen müssen (vgl. Hix/Hartson 1993, 38; Lin/Choong/Salvendy 1997, 271). Dies kann einerseits dadurch erreicht werden, dass Systeme den direkten Erwartungen der Nutzer an Funktionalität und Benutzungsgefühl entsprechen. Andererseits können Bedienoberflächen mentale Modelle aufgreifen, die der Anwender nicht im Bereich der Softwarenutzung kennt. Dies wird dann als das Verwenden von Metaphern bezeichnet.

Da heute bereits eine Vielzahl an Software-Applikationen besteht, haben Nutzer bestimmte bewusste und unbewusste Erwartungen an die Benutzung bestimmter

Softwaretypen. Bestehen gleiche Erwartungen bei einer Vielzahl an Nutzern, wird auch von Konventionen gesprochen. Solche Konventionen entstehen daraus, dass viele Applikationen oder einzelne, sehr populäre Applikationen bestimmte gestalterische Vorgehensweisen konsistent durchsetzen (siehe hierzu auch Kapitel 4.8). Verhält sich eine Software an bestimmten Stellen nicht so wie erwartet, kann dies Nutzer stark verwirren (Hix/Hartson 1993, 35). Dabei ist zu beachten, dass die Erwartungen der Nutzer stets schwerer wiegen als ein eindeutiges Design: selbst, wenn eine Benutzeroberfläche Funktionen und Bedienelemente mit logischen Namen und Bildern versieht, kann es zu Verwirrung kommen, wenn diese Funktionen überall sonst anders gestaltet werden (Ziefle 2002, 310). Der Nutzer muss stets als eine Ansammlung gemachter Erfahrungen betrachtet werden. Kaum ein Anwender wird die Anwendung als völliger Software-Neuling angehen (Blair-Early/Zender 2008, 101).

Das Erfüllen von Nutzererwartungen wird empfohlen, um die Effizienz der Nutzung zu steigern. Neben dem Begriff Erwartungskonformität wird auch von Kompatibilität mit den Nutzererwartungen gesprochen (vgl. Scapin/Bastien 1997, 227). Das Gedächtnis des Nutzers soll wie zuvor beschrieben nicht übermäßig belastet werden. Der Nutzer soll Elemente wiedererkennen können und sich nicht an sie erinnern müssen (vgl. Hix/Hartson 1993, 36ff.; Lidwell/Holden/Butler 2003, 200). Wenn also davon ausgegangen werden kann, dass Nutzer bestimmte Erwartungen an die Bedienung einer Software haben, sollte die Software diese Erwartungen bedienen. Dazu ist nicht nur die genaue Kenntnis der Nutzer notwendig, sondern auch die Beschaffenheit der Best Practices, der populärsten Vertreter der betrachteten Software-Gattung (vgl. Jordan 1998, 26f.). Eine Benutzeroberfläche muss nicht alle existierenden Konventionen berücksichtigen, sie sollte jedoch nur mit Vorsicht gegen sie verstoßen. Das Nicht-Einhalten einer Konvention sollte ein bestimmtes Problem lösen oder einen anderen Vorteil mit sich bringen. Ansonsten sollte versucht werden, Bedienungskonventionen einzuhalten (Blair-Early/Zender 2008, 100).

EK-01: Beachten von Gestaltungs-Best Practices
Wenn für die Gestaltung einer Funktion Best Practices existieren, soll die Applikation diese Best Practices einhalten.

Zu dem Einhalten von Nutzererwartungen und Konventionen gehören nicht nur die Modelle in der Vorstellung der Anwender, sondern auch die von diesen verwendete Technik. Je nachdem, ob klassische Computer, Tablet-PCs oder Smartphones bedient

werden sollen, ergibt sich eine bestimmte Bandbreite an Systemressourcen, Betriebssystemen und Zusatzprogrammen wie Browser oder Plug-ins, die bei der Einführung einer Software bedient werden sollten. Nutzer erwarten, dass eine Software auf ihrem Endgerät funktioniert. Dieser durchaus funktionale Aspekt ist bei der Analyse der Ziel-Anwender und dem anschließenden Gestalten zu beachten (vgl. Kaasinen 2005, 100f.). Um diese Anforderung zu berücksichtigen, empfiehlt sich eine Anwendung von Anforderungsmuster CO-04: *Konsistente Darstellung auf verschiedenen Endgeräten*.

4.12 Metaphern

Metaphern werden im Software- und Webdesign seit Jahrzehnten erfolgreich eingesetzt, um mentale Modelle des Anwenders aufzugreifen und so Benutzbarkeit zu verbessern. Oft treten mehrere Metaphern gleichzeitig auf, die sich gegenseitig ergänzen. Ihrer Beliebtheit entsprechend werden sie in Fachliteratur im Rahmen von Usability-Empfehlungen oft genannt (Hsu/Boling 2007, 209). Wenngleich Metaphern eine Form der Problemlösung darstellen, soll der Vollständigkeit wegen ihre Rolle innerhalb des Usability-Instrumentariums umrissen werden.

Bei Softwaredesign wird von Metaphern gesprochen, wenn Teile eines Systems eine Versinnbildlichung erhalten, die den Nutzern vertraut ist. Ein oft genanntes Beispiel ist die Schreibtisch-Metapher von Computern, bei der das Verwalten von Dateien ähnlich dargestellt ist wie das reale Bewegen von Blättern und Ordnern. Metaphern übertragen die Eigenschaften des verwendeten Sinnbildes auf die Funktionalität der Software. Dies hat den Vorteil, dass dem Nutzer neue Funktionsweisen und Abläufe nicht aufwendig erklärt werden müssen, da er das verwendete Sinnbild bereits kennt und übertragen kann (Stapelkamp 2010, 54). Gute Metaphern tragen zu einer höheren Usability bei, da sie die Distanz zwischen der Gedankenwelt des Anwenders und dem eigentlich abstrakten System verringern. Dadurch wird die kognitive Belastung gemindert und die Benutzung erleichtert (vgl. Bennet/Flach 2011, 113f.). Einfache Beispiele hierfür sind kleine Bilder-Icons auf Bedienoberflächen, die reale Objekte äußerlich imitieren und so deren Eigenschaften aufnehmen (z. B. der Papierkorb auf Desktop-PCs, das Disketten-Symbol zum Abspeichern, etc.). Bei vielen verwendeten Motiven ist die Grenze zwischen Metaphern und sog. „Imitationen“ fließend. Es können ganze Bedienoberflächen so gestaltet werden, dass sie wie eine andere Umgebung aussehen (z.B. Desktop-Metapher). Möglich ist auch, dass das System das

Verhalten einer bestimmten Sache imitiert. Dies ist beispielsweise bei computergesteuerten Assistenten der Fall, die wie Personen gestaltet sind und die Rolle von Beratern übernehmen wollen. Schlussendlich kann ein Design Funktionsweisen imitieren, beispielsweise kann ein Programm eine Tastatur darstellen und damit die Eingabe einer echten Tastatur imitieren (vgl. Lidwell/Holden/Butler 2003, 156).

Ein mit Funktions-Imitation verwandtes Instrument ist das Schaffen sog. „Affordance". Dabei handelt es sich um Bedienelemente, die physisch und sensorisch so gestaltet sind, dass sie sich für eine bestimmte Funktion gut zu eignen scheinen. Als Beispiel realer Objekte sind Türen zu nennen: Türen mit einem Griff laden dazu ein, aufgezogen zu werden, während Türen ohne Griff o. ä. ohne nachzudenken aufgedrückt werden. Dieses Prinzip lässt sich auf Bildschirmen einsetzen, um bestimmtes Nutzer-Verhalten zu erreichen. Ein plastisch gestalteter Knopf löst bei Nutzern den Wunsch aus, ihn anzuklicken um ihn so zu „drücken". Affordance wird geschaffen, wenn Metaphern oder Imitationen eindeutige Funktionsweisen aus der Realität aufgreifen und den Nutzer so zu einem bestimmten Verhalten bewegen (vgl. Lidwell/Holden/Butler 2003, 22f.).

Eine passende Metapher für ein bestimmtes System zu finden, ist selten einfach. So kann eine Metapher zwar Verständnis für einzelne Funktionsweisen schaffen, jedoch lässt sich mit ihr nie die komplette Funktionalität einer Software versinnbildlichen. Ob eine Metapher funktionieren kann, hängt in erster Linie von der Zielgruppe ab, von deren Bedürfnissen, Fähigkeiten und mentalen Modellen. So kommt es aufgrund fehlender Nutzerkenntnis oft zum unnötigen oder falschen Gebrauch von Metaphern (Stapelkamp 2010, 54ff.). Wenn eine Metapher allein die u. U. sehr komplexe Bedienoberfläche nicht vollständig versinnbildlichen kann, empfiehlt sich der Gebrauch mehrerer Metaphern in Kombination (Hsu/Boling 2007, 210f.). Die Auswahl geeigneter Metaphern erfolgt nach Kriterien wie Bekanntheit bei den Nutzern, Ähnlichkeit zwischen den Metaphern und deren visueller Repräsentation (vgl. Hsu/Boling 2007, 212) Wann eine Metapher überhaupt notwendig ist, lässt sich nicht pauschal an einigen wenigen Faktoren festmachen. Ihr Einsatz ist empfehlenswert, wenn eine Benutzeroberfläche von den Anwendern als so komplex und abstrakt wahrgenommen werden würde, dass eine unmittelbare Verständnis-Hilfe notwendig wäre (vgl. Bennet/Flach 2011, 120f; Blair-Early/Zender 2008, 104). Oft werden Metaphern auch ohne diesen dringenden Erklärungsbedarf verwendet. Da auf

Bildschirmen vielfältige Darstellungsmöglichkeiten gegeben sind, besteht die Versuchung, viele scheinbar veranschaulichende Bildelemente zu verwenden, um eine Benutzeroberfläche zu schmücken (vgl. Bennet/Flach 2011, 164f). Stapelkamp argumentiert, dass bloße Imitationen kein Ersatz für durchdachte Metaphern sind. Wichtiger ist stattdessen, sich ausreichend mit den Nutzerbedürfnissen zu beschäftigen (2010, 57).

Da Metaphern eine komplexe Lösungsvariante zum Aufgreifen mentaler Modelle des Nutzers sind, lassen sich für sie keine gesonderten Anforderungsmuster ableiten. Es soll hier lediglich der Hinweis erfolgen, dass bei Bedienoberflächen mit sehr neuartigen, abstrakten Bedienabläufen die mentalen Modelle der Anwender berücksichtigt werden sollten. Mit deren Hilfe können die Interface-Designer Gestaltungsvarianten finden, die für die Nutzergruppe intuitiv verständlich sind und keiner großen Erklärung bedürfen.

4.13 Schönheit und Ästhetik

Die Schönheit eines Designs wirkt sich auf dessen wahrgenommene Nutzungsfreundlichkeit aus. Nutzer glauben, dass ansprechende Designs eine höhere Benutzbarkeit haben als weniger ansprechende. Das gilt auch, wenn die tatsächliche Usability nicht besonders hoch ist. Schöne Designs laden den Anwender zur Benutzung ein, werden schneller akzeptiert und wirken einfach. Dies rührt daher, dass der erste Eindruck die Einstellung eines Menschen bezüglich einer Sache lange beeinflusst. Ansprechende Designs erzeugen eine positive Einstellung, machen den Nutzer bei Designfehlern geduldig und tragen zu einer besseren Langzeit-Usability bei (vgl. Tractinsky 1997, 101; Lidwell/Holden/Butler 2003, 20). Schönheit und Ästhetik hängen mit dem subjektiven Empfinden der Nutzer zusammen. Die Empfehlung lautet daher, ein Design zu schaffen, das von den anvisierten Zielnutzern als ansprechend oder wenigstens angenehm empfunden wird (Ludewig/Lichter 2010, 371; Kurosu/Kashimura 1995, 293). Wenngleich dies eine Grundanforderung an jedes System mit guter Usability darstellt, lassen sich aufgrund der vielen subjektiven Faktoren, welche die wahrgenommene Schönheit bestimmen, keine überprüfbaren Anforderungstexte für diesen Bereich formulieren. Um die geschaffene Ästhetik eines Designs zu überprüfen, sollten im Rahmen der Usability-Evaluation Fragen zur wahrgenommenen Schönheit gestellt werden.

4.14 Lesbarkeit

Lesbarkeit beschäftigt sich mit der Präsentation von Text auf Bildschirmen und den Faktoren, die das Erfassen des Textes durch den Nutzer fördern. Die Leistung des Nutzers bei der Arbeit mit einem System verbessert sich, wenn die Informationsdarstellung dessen Art der Informationsaufnahme berücksichtigt (vgl. Scapin/Bastien 1997, 223f.).

Grundsätzlich gilt, dass Informationen möglichst klar dargestellt werden sollten. Sie sollten schnell erfasst werden können, ohne dass dabei Verwirrung entsteht. Für Bildschirmtext gilt daher, dass dieser gut lesbar gestaltet ist. Ein Text ist lesbar, wenn „die Information der einzelnen Zeichen in leserlich angeordneter Zeichenfolge erfass[t] und verstanden [werden] können“ (Böhringer/Bühler/Schlaich 2011, 28). Für gut lesbare Texte gibt es eine Reihe von Gestaltungsmöglichkeiten: Buchstaben und Zeichen sollten groß genug geschrieben sein und die Textmenge sollte in einem angemessenen Verhältnis zur Darstellungsfläche stehen. Farben sollten effektiv zur Lesbarkeit beitragen, wobei empfohlen wird, kräftige Farben für Schrift und blasse Farben für den Hintergrund zu wählen. In jedem Fall gilt ein hoher Kontrast zwischen Schrift und Hintergrund als gut lesbar. Der Kontrast sollte nicht absolut sein und es sollten keine Komplementärfarben gewählt werden, da diese am Bildschirm anstrengend zu lesen sind. Bei alledem ist auch zu beachten, aus welcher Entfernung der Nutzer den Bildschirm betrachtet und welche Größe dieser hat[2]. Aufgrund der vielen Gestaltungsmöglichkeiten gibt es wenige absolute Empfehlungen für die Schriftgestaltung an Bildschirmen. Was genau die beste Ausgestaltung für ein System ist, wird von Gestaltern entschieden und hängt u.a. mit den technischen Möglichkeiten und Einschränkungen des Mediums und dem Corporate Design eventueller Marken zusammen. Je nach Geeignetheit können eine oder mehrere der folgenden Anforderungen zu lesbarem Text gewählt werden (vgl. Böhringer/Bühler/Schlaich 2011, 525).

LE-01: Lesbarer Text
Die Applikation soll für Fließtext einen Schriftgrad zwischen 9 und 12 Pixel verwenden.

[2] vgl. Böhringer/Bühler/Schlaich 2011, 525; Jordan 1998, 33; Preece 2000, 281; Scapin/Bastien 1997, 223 f; Lidwell/Holden/Butler 2003, 148

Die Applikation soll Schrift mit einem hohen, aber nicht absoluten Farbkontrast gegenüber dem Hintergrund darstellen.

Die Applikation soll eine Schrift verwenden, die auf allen Betriebssystemen verfügbar ist.

Gute Lesbarkeit geht über die reine Darstellung der Schrift hinaus. Es sollten Formulierungen verwendet werden, die die Nutzer verstehen. Lesbarkeit wird u. a. von Wortlängen, Satzlängen, der Anzahl von Satzteilen sowie der Anzahl von Silben in einem Satz beeinflusst. Auch hier gibt es Gestaltungsempfehlungen wie das Vermeiden unnötiger Wörter oder das Verwenden aktiver Satzkonstruktionen (Lidwell/Holden/Butler 2003, 198). Technische Begriffe oder system-orientierte Formulierungen sollten vermieden werden. Ein Design sollte den Sprachgebrauch und das Vokabular der anvisierten Nutzergruppe verwenden, um seine Informationen effektiv zu vermitteln (vgl. Hix/Hartson 1993, 41; Nielsen/Mack 1994, 30). Diese Anforderung setzt voraus, dass der Sprachgebrauch der Endnutzer bekannt ist.

LE-02: Verständlicher Text
Zeigt die Applikation Informationen in Form von Text an, soll die Applikation Formulierungen verwenden, die die Nutzer verstehen.

Wird eine Variante dieser Anforderungen verwendet, sollte im Vorhinein festgelegt werden, wie die Lesbarkeit später überprüft wird. Hier empfiehlt sich einmal mehr das Testen der Lesbarkeit durch potentielle Endnutzer, da deren Eigenschaften über Lesegeschwindigkeit und Textverständnis entscheiden (vgl. Scapin/Bastien 1997, 223f.).

4.15 Hilfe anbieten

Unabhängig von dem Design der Bedienoberfläche ist es bei vielen Software-Applikationen empfehlenswert, eine Hilfe oder eine Bedienungsanleitung einzubauen (Rampl 2007). Zunächst sollte immer versucht werden, die Bedienung so benutzbar und einfach wie möglich zu gestalten. In den meisten Anwendungen, die über den Funktionsumfang einer einfachen Informations-Webseite hinaus gehen, wird es jedoch zu Funktionen kommen, die gelernt und erklärt werden müssen. In diesem Fall sollte die Software eine Hilfe oder Bedienungsanleitung anbieten, die v.a. die komplexen Funktionen der Software behandelt. Sie sollte einfach zu lesen und zu durchsuchen

sein. Bedienungsanleitungen werden üblicherweise nicht vor der Benutzung gelesen, sondern erst dann, wenn ein dringendes Problem aufgetreten ist. Die Hilfestellung sollte deshalb stets aufrufbar, der Hinweis zu ihr jedoch nur dezent sichtbar sein (vgl. Blair-Early/Zender 2008, 102; Nielsen 1993, 148ff.).

Eine Hilfestellung kann verschiedene Formen haben. Die einfachste Form ist eine Bedienungsanleitung aus Text und ggf. Bildern. Diese kann direkt in die Software eingebaut werden, auf einer Internetseite bereit stehen oder als Druckversion vorliegen (vgl. Nielsen 1993, 148f.). Möglich sind auch Erklärungen, die direkt neben einer Funktion eingeblendet werden, eingeblendete Tipps, Beispiele oder Tutoriale (vgl. Xie 2002, 910).

HI-01: Vorliegen einer Hilfestellung
Die Applikation soll eine Hilfestellung anbieten.

Möglich sind Hinweise, die den Nutzer während der Benutzung begleiten oder ihn sogar durch die Benutzung führen (vgl. Scapin/Bastien 1997, 226). In den Kapitel zu Kontrolle und Indivisualisierbarkeit wurde bereits auf den Unterschied zwischen Anfängern und Experten hingewiesen und dass für Anfänger eine solche begleitende Hilfestellung angeboten werden kann.

HI-02: Komplexe Funktionen mit Hinweisen begleiten
Die Applikation soll dem Nutzer bei Verwendung einer Funktion eine begleitende Hilfestellung anbieten.

Werden begleitende Hinweise für unerfahrene Nutzer angeboten, sollten erfahrene Nutzer die Möglichkeit bekommen, die Hinweise zu überspringen, damit sie nicht verlangsamt werden (vgl. Scapin/Bastien 1997, 226; Lin/Choong/Salvendy 1997, 271; Nielsen 1993, 41).

HI-03: Hilfestellung überspringen
Bietet eine Funktion Hilfestellung, sollen Nutzer die Möglichkeit haben, diese Hilfestellung zu umgehen.

Das Vorhandensein einer Bedienungsanleitung oder Hilfestellung ist zu keinem Zeitpunkt ein Ersatz für benutzerfreundliches Design (vgl. Blair-Early/Zender 2008, 102). Die genannten Empfehlungen sind daher losgelöst von der Anwendung der anderen Usability-Anforderungsmuster zu sehen.

4.16 Überblick über die Usability-Anforderungsmuster

Zuvor wurden 15 Gestaltungsbereiche für Usability diskutiert, von denen 13 mindestens ein Anforderungsmuster erhielten. Die folgende Tabelle zeigt die 47 Usability-Anforderungsmuster im Überblick. Es ist zu erkennen, dass die Bereiche *Kontrolle und Einschränkung*, *Effizienz* und *Fehler-Handhabung* die meisten Muster erhalten haben. Gerade in *Kontrolle und Einschränkung* sowie *Fehler-Handhabung* werden viele Detailanforderungen gestellt, die dem Nutzer eine gute Steuerbarkeit des Systems verschaffen.

In Kapitel 2 wurden 7 Usability-Erfolgsmerkmale definiert, bei deren Vorliegen man von guter Usability spricht. Dies waren Effektivität und Effizienz der Nutzung, Zufriedenstellung, Lernbarkeit der Nutzung, Merkbarkeit der Funktionsweise, geringe Fehlerraten und geringe kognitive Belastung. Viele der genannten Anforderungsmuster unterstützen eines oder mehrere dieser Erfolgsmerkmale. Im Folgenden wird noch einmal zusammengefasst, wie die einzelnen Usability-Bereiche die verschiedenen Usability-Erfolgsmerkmale fördern.

- Muster zu Kontrolle und Einschränkung fördern vorrangig die Effektivität der Nutzung, die Lernbarkeit der Nutzung und geringe Fehlerraten.
- Muster aus dem Bereich Effizienz tragen nicht nur zu besserer Effizienz bei, sondern auch zu höherer Zufriedenstellung und geringer kognitiver Belastung.
- Die Empfehlungen zu Individualisierbarkeit verbessern v.a. die Effektivität, die Effizienz und die Zufriedenstellung der Nutzung.
- Muster zur Fehlerhandhabung senken nicht nur die Fehlerraten, sie erhöhen auch die Lernbarkeit und Merkbarkeit der Nutzungsweise und erhöhen damit Effektivität und Effizienz der Nutzung.
- Das Schaffen von Eintrittspunkten stärkt Effektivität und Effizienz der Nutzung, fördert Lernbarkeit und Merkbarkeit der Abläufe und verringert die Gefahr von Fehlern.
- Die Muster zu Wegfindung tragen zu den gleichen Usability-Zielen bei wie das Schaffen von Eintrittspunkten. Sie verringern außerdem die kognitive Belastung.
- Die richtige Anordnung von Informationen fördert eine effiziente Nutzung, eine geringe kognitive Belastung und eine bessere Lernbarkeit.

- Die Muster zu Konsistenz stärken Lernbarkeit und Merkbarkeit der Nutzung.
- Die Sichtbarkeit des Systemstatus verstärkt ebenfalls die Lernbarkeit und verringert die Gefahr von Fehlern.
- Muster zu Rückmeldung verbessern die Zufriedenstellung, die Effektivität, die Lern- und Merkbarkeit der Nutzung und senken das Risiko von Fehlern.
- Das Beachten von Erwartungskonformität stärkt die Zufriedenstellung und die Lernbarkeit des Systems.
- Eine gute Lesbarkeit fördert Effektivität, Effizienz, Zufriedenstellung, Lernbarkeit und geringe Fehlerraten.
- Das Anbieten von Hilfe stärkt ebenfalls die Lernbarkeit, fördert die Effektivität der Nutzung und senkt mittelfristig die Fehlerraten.

In Kapitel 4.1 wurden die standardisierten Anforderungen als isolierte Empfehlungen formuliert. In den Anforderungsmustern in Anhang A werden zusätzlich Abhängigkeiten, Verknüpfungen und Konflikte zwischen den Mustern angeführt. In den Abhängigkeiten eines Anforderungsmusters werden jene Anforderungsmuster genannt, die in Kombination miteinander umgesetzt werden sollten. Verknüpfungen beschreiben jene Muster, die gleiche oder ähnliche Ziele haben, ohne dass diese jedoch zwingend gemeinsam umgesetzt werden sollten. Muster, die miteinander in Konflikt stehen, bringen Anforderungen hervor, die nur schwer gleichzeitig umgesetzt werden können. Eine vollständige Gegenüberstellung aller Muster findet sich in Tabelle 5. Darin sind Abhängigkeiten mit einem „A“, Verknüpfungen mit einem „V“ und Konflikte mit „K“ markiert.

Kontrolle und Einschränkung	
KO-01	Explizite Aktionskontrolle
KO-02	Steuerbarkeit durch Vorliegen einer Navigation
KO-03	Funktionen pausieren und abbrechen
KO-04	Informationen zu Vor-Einstellung
KO-05	Vor-Einstellung abspeichern
KO-06	Zwei Wege der Aufgabenbewältigung
KO-07	Kein Anbieten nicht verfügbarer Funktionen
Effizienz	
EF-01	Inhalt herausstellen
EF-02	Aufwand des Nutzers minimieren
EF-03	Körperlichen Einsatz minimieren
EF-04	Verständliche Anweisungen
EF-05	Aufteilen komplexer Aufgaben
EF-06	Übersichtlichkeit
EF-07	Kein Merken-Müssen von Informationen
Individualisierbarkeit	
IN-01	Aktive Personalisierung anbieten
IN-02	Passive Personalisierung anbieten
IN-03	Mehrere Aktionen gleichzeitig ausführen
Fehler-Handhabung	
FE-01	Bestätigung kritischer Aktionen
FE-02	Fehler-Benachrichtigung
FE-03	Aktionen rückgängig machen
FE-04	Isoliertes Korrigieren von falschen Eingaben
FE-05	Überprüfen von Eingaben vor Auslösen der Funktion
FE-06	Automatisches Speichern überschriebener Inhalte
Eintrittspunkte	
EP-01	Startpunkt anbieten
EP-02	Jederzeitiges Zurückkehren zum Startpunkt
EP-03	Keine Barrieren im Startpunkt
EP-04	Überblick-Funktion des Startpunkts
Wegfindung	
WE-01	Orientierung anbieten
WE-02	Aussagekräftige Bedienelemente
Anordnung von Informationen	
AI-01	Wichtige Informationen hervorheben
AI-02	Informationen gruppieren
AI-03	Ständig erreichbare Navigation
Konsistenz	
CO-01	Alle Teile einer Applikation ähnlich gestalten
CO-02	Konsistent gestaltete Funktionen
CO-03	Konsistent gestaltete Informationen
CO-04	Konsistente Darstellung auf verschiedenen Endgeräten
Sichtbarkeit des Systemstatus	
SI-01	Systemstatus anzeigen
Rückmeldung	
RM-01	Rückmeldung bei Aktionen
RM-02	Hinweis bei langer Bearbeitung
RM-03	Sofortiges Anzeigen von Eingaben
RM-04	Erfolgreiche Bearbeitung anzeigen
Erwartungskonformität	
EK-01	Beachten von Gestaltungs-Best Practices
Lesbarkeit	
LE-01	Lesbarer Text
LE-02	Verständlicher Text
Hilfe anbieten	
HI-01	Vorliegen einer Hilfestellung
HI-02	Komplexe Funktionen mit Hilfe begleiten
HI-03	Hilfestellung überspringen

Tabelle 4: Übersicht der Usability-Anforderungsmuster
Quelle: Eigene Darstellung

Die meisten Abhängigkeiten zwischen den Mustern bestehen dann, wenn deren Anforderungen aus dem gleichen Bereich stammen. Das rührt daher, dass innerhalb eines Bereiches einige Anforderungen aufeinander aufbauen und sich mit den gleichen Aspekten der Bedienoberfläche befassen. Beispiele sind das Aufstellen einer Navigation im Bereich *Kontrolle und Einschränkung* oder das Anbieten und Gestalten eines Startpunkts im Bereich *Eintrittspunkte*. Der Bereich *Kontrolle und Einschränkung* hat außerdem mehrere Abhängigkeiten mit den Bereichen *Individualisierbarkeit* und *Fehler-Handhabung*. Dies ist insofern verständlich, dass auch letztere Bereiche im weitesten Sinne die Kontrolle durch den Nutzer mitgestalten. So geben die Muster IN-01: *Aktive Personalisierung anbieten* und FE-03: *Aktionen rückgängig machen* dem Nutzer direkte Steuermöglichkeiten, wie sie auch im Bereich Kontrolle empfohlen werden.

In den Verknüpfungen der Muster deutet sich an, zu welchen Usability-Qualitäten die einzelnen Muster beitragen. Einerseits teilen sich viele Muster aus den jeweils gleichen Bereichen gemeinsame Ziele, was schlicht an der hier vorgenommenen Gruppierung der Muster liegt. Darüber hinaus sind die Verknüpfungen erkennbar, die in Kapitel 4.1 angesprochen wurden: Muster aus den Bereichen *Lesbarkeit* und *Hilfestellung* unterstützen die Effizienz der Benutzung; Muster zu *Rückmeldungen bei Aktionen* und das Anzeigen ausgewählter Informationen tragen zu einem besseren Lernen und Verstehen der Applikation bei. Ausgeprägte Steuerungsmöglichkeiten, wie sie im Bereich *Kontrolle und Einschränkungen* beschrieben sind, tragen ebenso zu einer guten Fehlerhandhabung bei wie die Anforderungen zu eben diesem Bereich. Es zeigt sich, dass eine einzelne Anforderung nicht nur auf ein Usability-Merkmal hinarbeiten kann, sondern auf mehrere. Das macht die hier getroffene Gruppierung der Muster willkürlich. Eine Einteilung könnte ebenso gut nach Usability-Zielen oder nach Funktionsbereichen erfolgen. Wichtiger als die Gruppierung der Muster erscheint der Überblick über alle möglichen Anforderungen, da dieser zu einer ganzheitlichen Betrachtung der Usability-Potentiale und -Herausforderungen einer Software führt. Wie in den zu Anfang genannten Usability-Definitionen beschrieben, ist Usability ein Potenzial, das durch mehrere erfüllte Anforderungen wächst.

	KO-01	KO-02	KO-03	KO-04	KO-05	KO-06	KO-07	EF-01	EF-02	EF-03	EF-04	EF-05	EF-06	EF-07	IN-01	IN-02	IN-03
KO-01			V				K		K								
KO-02			V														
KO-03	V	A, V															
KO-04					A												
KO-05				A											A, V		
KO-06												V					V
KO-07																	
EF-01																	
EF-02	K									A, V	A, V						V
EF-03									A, V		A, V						V
EF-04									V	V							
EF-05						V								A			
EF-06														A			
EF-07													A				
IN-01				A	A, V											V	
IN-02				A	A										V		
IN-03						V			V	V							
FE-01										K							
FE-02										A							
FE-03		A	A, V														
FE-04																	
FE-05				A, V		V											
FE-06	K																
EP-01		A															
EP-02																	
EP-03																	
EP-04																	
WE-01																	
WE-02																	
AI-01							V	A, V					V				
AI-02																	
AI-03		A															
CO-01																	
CO-02																	
CO-03																	
CO-04																	
SI-01				V													
RM-01																	
RM-02																	
RM-03																	
RM-04																	
EK-01																	
LE-01											V						
LE-02											V						
HI-01											V			A			A
HI-02						V					A, V						A
HI-03																	V

A = Abhängigkeiten, V = Verknüpfungen, K = Konflikte

	FE-01	FE-02	FE-03	FE-04	FE-05	FE-06	EP-01	EP-02	EP-03	EP-04	WE-01	WE-02	AI-01	AI-02	AI-03	CO-01	CO-02
KO-01						K											
KO-02				A											A		
KO-03			V														
KO-04					V	A											
KO-05																	
KO-06																	
KO-07													V				
EF-01													A, V				
EF-02																	
EF-03	K																
EF-04																	
EF-05																	
EF-06													A, V				
EF-07																	
IN-01																	
IN-02																	
IN-03																	
FE-01																	
FE-02																	
FE-03		A		V	A, V												
FE-04		A	A, V		V												
FE-05			V	A, V													
FE-06																	
EP-01								A	A		V						
EP-02							A										
EP-03							A										
EP-04							A		A, V		V		A, V				
WE-01							A, V	A		A, V		A					
WE-02											A						A, V
AI-01									V	V				V			
AI-02													A, V				
AI-03																	
CO-01																	A, V
CO-02												A, V				A, V	
CO-03												A, V				A, V	A, V
CO-04																A, V	A, V
SI-01																	
RM-01																	
RM-02																	
RM-03		A			A												
RM-04		A, V	A, V														
EK-01																	
LE-01																	
LE-02																	
HI-01																	
HI-02																	
HI-03																	

A = Abhängigkeiten, V = Verknüpfungen, K = Konflikte

	CO-03	CO-04	SI-01	RM-01	RM-02	RM-03	RM-04	EK-01	LE-01	LE-02	HI-01	HI-02	HI-03
KO-01													
KO-02													
KO-03													
KO-04			V										
KO-05													
KO-06												A, V	
KO-07													
EF-01													
EF-02													
EF-03													
EF-04									A, V	A, V	A, V	A, V	
EF-05													
EF-06			A										
EF-07													
IN-01													
IN-02													
IN-03													V
FE-01													
FE-02				A			V						
FE-03							V						
FE-04													
FE-05													
FE-06													
EP-01													
EP-02													
EP-03													
EP-04													
WE-01													
WE-02	A, V												
AI-01													
AI-02													
AI-03													
CO-01	V	V							A				
CO-02	V	V											
CO-03		V							A				
CO-04	A, V							A, V	A				
SI-01				A, V	A, V	V							
RM-01			A, V		A, V	V							
RM-02			V	A, V		V							
RM-03			A, V	A, V	V								
RM-04													
EK-01		A, V											
LE-01	A	A								A, V			
LE-02									A				
HI-01												V	
HI-02											A, V		
HI-03											A	A	

A = Abhängigkeiten, V = Verknüpfungen, K = Konflikte

Tabelle 5: Abhängigkeiten, Verknüpfungen und Konflikte zwischen den Usability-Anforderungsmustern
Quelle: Eigene Darstellung

Bei der Auswahl von Anforderungen für ein System ist darauf zu achten, keine sich widersprechenden Vorgaben aufzustellen. Deshalb müssen die Anforderungsmuster, die miteinander in Konflikt stehen, sehr sorgfältig ausformuliert werden. Die Zahl der in diesem Katalog in Konflikt stehenden Muster ist eher gering. KO-01: *Explizite Aktionskontrolle* besagt, die Applikation solle nur jene Funktionen ausführen, die eine Entscheidung des Nutzers erfordern. Dies widerspricht sich mit den Mustern, die dem Nutzer mit automatischen Funktionen Bequemlichkeit oder Sicherheit anbieten wollen, wie beispielsweise EF-02: *Aufwand des Nutzers minimieren* und FE-06: *Automatische Speichern überschriebener Inhalte*. Dieser Konflikt kann damit gelöst werden, dass dem Nutzer automatisch ablaufende Funktionen entweder erklärt oder gar nicht angezeigt werden. Der Konflikt zwischen EF-03: *Körperlichen Einsatz* minimieren und FE-01: *Bestätigung kritischer Aktionen* beschreibt das Spannungsfeld zwischen schneller, flüssiger Benutzung und dem ausführlichen Erklären der Funktionen. Der Konflikt wird dadurch gelöst, dass man die verlangsamende Bestätigung nur bei wirklich kritischen Funktionen verwendet.

In den 47 Mustern existieren 3 potenzielle Konflikte. Gleichzeitig bestehen viele Überein-stimmungen in den Zielen der Muster sowie mehrere Kombinationsmöglichkeiten. Die Anwendung des Katalogs sollte daher gut geeignet sein, bestehende funktionale Anforderungskataloge zu ergänzen, ohne gleichzeitig neue Komplikationen zu schaffen.

5 Expertenbefragung zur Beurteilung der Usability-Anforderungsmuster

Die zuvor erarbeiteten Anforderungsmuster wurden im Rahmen von Gesprächen mit potentiellen Anwendern auf ihre Qualität hin geprüft. Im Folgenden werden die Gesprächsergebnisse zusammengefasst dargestellt.

Es fanden Gespräche mit drei Praktikern statt, die im Folgenden A, B und C genannt werden:

- A ist der Geschäftsführer eines Softwarebüros mit fünf Mitarbeitern, das Webseiten, Smartphone-Applikationen und PC-Software gestaltet, entwickelt und umsetzt. Das Unternehmen ist seit acht Jahren erfolgreich am Markt und konzentriert sich auf die Schaffung moderner, modischer Designs.
- B ist der Geschäftsführer eines Softwarebüros mit sieben Mitarbeitern, von denen vier als Programmierer arbeiten. Das Unternehmen stellt seit über zehn Jahren erfolgreich individuelle Web-anwendungen her. Dabei setzt das Unternehmen ein detailliertes Anforderungsmanagement ein.
- C ist der Requirements-Ingenieur eines Programmierbüros mit 13 Mitarbeitern. Das Unternehmen bearbeitet sowohl individuelle Kundenaufträge als auch ein eigenes Softwareprodukt zur Heizkostenabrechnung für Hausbesitzer. Die gängigsten Produkte sind Datenbank- und Systemanwendungen. Das Unternehmen ist seit 1984 erfolgreich am Markt und arbeitet im Kundenkontakt und in der Softwareentwicklung konsequent mit Anforderungen.

Bei allen drei Unternehmen sind die zuvor genannten Kriterien für die Befragung erfüllt. Alle pflegen ein Anforderungsmanagement, wenngleich dies bei jedem Unternehmen unterschiedlich stark ausgeprägt ist. Sie entwickeln außerdem eigene Software-Produkte, bei denen Usability eine entscheidende Rolle spielt. Die Gespräche folgten im Wesentlichen dem in Kapitel 3.2 vorgestellten Fragenkatalog. Ziel war es, die Qualität der Anforderungsmuster zu erfragen, deren praktische Anwendbarkeit zu beurteilen und Verbesserungspotenziale aufzudecken.

5.1 Ergebnisse der Expertenbefragung

Die Gespräche begannen mit einigen allgemeinen Fragen zum Anforderungsmanagement der Unternehmen.

A.a. Arbeiten Sie bei jedem Projekt mit Anforderungen?

B und C setzen bei jedem Auftrag Anforderungsmanagement ein, A nur bei manchen. Alle Befragten beschrieben den Umstand, dass Kunden so gut wie nie in der Lage seien, ihre Wünsche in Form eines brauchbaren Lastenheftes zu formulieren. Es finden meist Beratungsgespräche statt, deren Ergebnis schließlich ein Pflichtenheft mit genauen Anforderungen ist. C antwortete: „Das Aufstellen von Anforderungen ist unverzichtbar. Die Anforderungen sind immer Verhandlungsgegenstand und müssen deshalb immer sorgfältig ausgearbeitet sein. Komplexe Modellierungswerkzeuge kommen allerdings erst ab einer bestimmten Projektgröße zum Einsatz."

A.b Wie viele Anforderungen hat ein Projekt bei Ihnen im Durchschnitt?

Zu dieser Frage konnten keine genauen Antworten gegeben werden. Während in der Literatur Zahlen wie ca. 50 - 100 Anforderungen pro Projekt genannt werden (vgl. Ebert 2012, S. 99), variiert die Zahl in der Realität je nach Detailgrad der Anforderungen. B beschrieb, dass bereits eine einzelne Funktion einer Software bis zu 50 Anforderungen haben kann.

A.c Arbeiten Sie mit Anforderungsmustern?

Diese Frage wurde von allen Befragten verneint. B und C begründeten dies damit, dass ihre Kundenaufträge so individuell seien, dass sich Anforderungen nur selten wiederholten. Das Verwenden von Vorlagen oder Mustern habe daher wenig Sinn.

A.d Berücksichtigen Sie Usability in der Softwareentwicklung?

Diese Frage bejahten alle Befragten. In allen drei Unternehmen wird Usability während der gesamten Entwicklungsphase von mehreren beteiligten Mitarbeitern berücksichtigt. C fasste zusammen: „Ja, Usability ist für unsere Produkte unverzichtbar, weil sie akzeptanzentscheidend ist." Bei keinem der Unternehmen finden jedoch regelmäßige Usability-Tests statt, viele notwendige Veränderungen entstehen stattdessen aufgrund von Kundenrückmeldungen.

A.e Verwenden Sie Usability-Anforderungen?

Keines der Unternehmen stellt eigene Usability-Anforderungen auf. Usability fließt permanent bei allen Befragten während der Entwicklung mit ein.

B.a Wie beurteilen Sie die Gliederung der Usability-Anforderungsmuster? Ist die Reihenfolge sinnvoll?

Hierzu gab es gemischte Meinungen. C empfand die Gliederung als sinnvoll. B konnte sich nicht vorstellen, mit der Gliederung zu arbeiten, da bei B die Anforderungen stets nach Features gruppiert sind. A schlug vor, die Muster nach Arbeitsphasen zu sortieren. Einige Usability-Bereiche, wie zum Beispiel Texte oder Hilfestellungen, würden erst gegen Ende der Entwicklungszeit bearbeitet und seien beim anfänglichen Aufbau der Systemarchitektur noch nicht relevant.

B.b Können Sie bestimmte Muster gut finden, wenn Sie an ein bestimmtes Problem denken?

B argumentierte, dass alle Muster sich mit dem gleichen Problem beschäftigten. A merkte an, dass die entsprechenden Probleme erst einmal bekannt sein müssten. Oft wisse man im Vorhinein nicht, welche Usability-Herausforderung an welcher Stelle lauere. C gelang es hingegen gut, bestimmte Anforderungsvorlagen auf Basis eines Problembereiches zu finden.

C.a Sind alle Elemente der Muster verständlich?

Alle Befragten verstanden die Elemente der Muster auf Anhieb. A merkte an, dass man die Muster in Verknüpfungen, Abhängigkeiten und Konflikte ausschreiben könne, um die Usability der Muster zu erhöhen.

C.b Sind alle notwendigen Informationen enthalten?

Für B und C waren die gegebenen Informationen ausreichend. A wünschte sich zusätzlich Gestaltungsempfehlungen, um sofort Lösungsvorschläge zu den Anforderungen griffbereit zu haben.

C.c Sind die Muster verständlich geschrieben/formuliert?

Auch hier gab es bei den Befragten keinen Diskussionsbedarf.

C.d Sind die Muster ausreichend problembezogen?

A und C empfanden die Auswahl an Mustern als vollständig genug, um durch ein Abarbeiten der Muster auf alle möglichen Usability-Probleme aufmerksam zu werden. B merkte hingegen an, dass ein gezieltes Auswählen der Muster nach Problembereich schwierig sei, da bei B die Muster weniger nach Problembezug, sondern nach Angemessenheit und Kosten-Nutzen-Verhältnis ausgewählt würden.

C.e Ist die Balance zwischen Konkretheit und Abstraktion angemessen?

B merkte an, dass die Muster sehr allgemein gehalten wären. B setze sehr feine, konkrete Muster ein. Für C entscheide sich der Detailgrad der Anforderungen im Ausgestalten der Parameter.

D.a Können die Muster/Anforderungen Ihnen dabei helfen, ein Design zu verbessern?

A bejahte dies, da die Muster sich sehr gut dazu eigneten, Designs zu überprüfen und Denkprozesse anzuregen. Für B und C sind die Muster inhaltlich selbstverständlich, da ihre Mitarbeiter einen hohen Erfahrungsstand hätten.

D.b Helfen die Muster bei der Problemlösung? Sind typische Usability-Probleme abgedeckt?

Für A und C sind alle typischen Usability-Bereiche abgedeckt.

D.c Schaffen die Muster eine gemeinsame Basis für die Kommunikation? Entspricht die Sprache der ihres Teams?

Auch hier bejahten alle Befragten die Angemessenheit der Begriffswahl, wenngleich bei einigen Begriffen englische Varianten verwendet würden.

D.d Enthalten die Muster für Sie relevantes Wissen?

Dies bejahten alle Befragten. Laut A eigneten sich die Muster gut, um ein standardisiertes Wissen über Usability für ein Unternehmen aufzubauen.

D.e Kann man die Muster gut anwenden? Was würden Sie verändern, um sie in Ihre Abläufe einzubringen?

Die erfahrenen Mitarbeiter von B und C berücksichtigen Usability bereits ohne die Verwendung von Anforderungen. C denkt, dass die Muster einen guten Leitfaden für unerfahrenere Entwickler darstellen. B argumentiert, dass das Inhaltsverzeichnis der Muster sich gut als Checkliste eigne, um eine Anforderungsspezifikation auf Vollständigkeit zu überprüfen. A wünscht sich den Katalog als Online-Checkliste mit kurzen, prägnanten Titeln in einer Übersicht, bei der auf Wunsch Gestaltungsbeispiele und Use Cases für jede mögliche Software-Form aufgerufen werden können.

E.a Stimmen Sie mit den Mustern inhaltlich überein oder haben Sie andere Erfahrungen oder Kenntnisse?

Bis auf wenige Ausnahmen, die im vorliegenden Muster-Katalog bereits berücksichtigt wurden, werden alle gängigen Usability-Empfehlungen genannt, die die Befragten bereits anwenden oder schon angewendet haben.

E.b Glauben Sie an die Nützlichkeit dieser Muster?

A hält Muster als solche nicht für nützlich und wünscht sich stattdessen die zuvor beschriebene Checkliste, um Usability bei der Gestaltung zu berücksichtigen. Für B sind die Muster irrelevant, weil das enthaltene Usability-Wissen bei seinen Mitarbeitern bereits vorliege. C will die Muster ebenfalls nicht als solche verwenden, die Empfehlungen aber in seine internen Richtlinien zur Softwareentwicklung aufnehmen.

5.2 Rückschlüsse aus den Befragungsergebnissen

Durch die Hinweise der Befragten wurden die erarbeiteten Anforderungsmuster mehrfach verändert. Einerseits wurden missverständliche Formulierungen geändert, andererseits wurden viele Vorlagentexte so formuliert, dass sie nun einen Prüfmechanismus erhalten. B wies explizit darauf hin, dass für sein Team keine Anforderungen in Frage kämen, deren Erfüllungsgrad diskutiert werden könne. Anforderungen müssen stets so formuliert sein, dass ihr Erfüllungsgrad genau geprüft werden kann. So wurde in einer älteren Version des Musters C0-02: *Konsistent gestaltete Funktionen* die Formulierung „(...) sollen die Funktionen überall ähnlich gestaltet sein“ verwendet. Diese wurde auf B’s Vorschlag hin geändert zu „(...) sollen Funktionen überall identisch formatiert sein“, da diese Formulierung überprüfbar sei. Änderungen dieser Art wurden an mehreren Mustern vorgenommen. Muster, bei

denen kein Prüfmechanismus gefunden werden konnte, wurden aus dem Katalog entfernt. Dazu zählte u. a. ein Muster zum Bereich *Schönheit und Ästhetik*, das die ansprechende Gestaltung der Bedienoberfläche forderte.

Ein Ziel der Befragung war es, die Qualität der erstellten Usability-Anforderungsmuster beurteilen zu lassen. Die befragten Praktiker haben eine Vielzahl an Hinweisen in Bezug auf die zuvor genannten Qualitätskriterien gegeben. Es muss jedoch berücksichtigt werden, dass keiner der Befragten Anforderungsmuster einsetzt, um Anforderungen zu schreiben. Dies hat einen erkennbaren Einfluss auf die gegebenen Antworten und stellt eine Limitation der Befragung dar. Die Frage nach der Qualität der Muster wurde außerdem dadurch beeinflusst, dass das Thema der Muster bereits von allen Befragten in anderer Form in den Entwicklungsprozess einfließt.

Bei der Prüfung der Auffindbarkeit der Muster wurden die unterschiedlichen Arbeitsweisen der Befragten deutlich. Je nachdem, in welchem Arbeitsschritt die Anwender bestimmte (Usability-)Anforderungen aufstellen und bearbeiten, wird die vorgeschlagene Reihenfolge der Muster akzeptiert oder abgelehnt. Unternehmen A, das sehr designorientiert und straff arbeitet, wünscht sich keine lange Liste mit Mustern, sondern eine schnell nutzbare Checkliste mit Lösungsvorschlägen. Unternehmen C, das ein viel stärker ausgeprägtes Requirements Engineering lebt, sieht den Muster-Katalog als Leitfaden und kann zielorientiert Anforderungsvorlagen finden und auswählen.

Die Verständlichkeit der Muster war größtenteils gegeben; kleinere Verbesserungsvorschläge wurden im vorliegenden Muster-Katalog berücksichtigt. Die Anforderungsmuster wurden grundsätzlich als hilfreich empfunden. Die Befragungsergebnisse lassen erkennen, dass Usability bei allen Befragten ein zentrales Thema darstellt. Dies deckt sich mit der entsprechenden Meinung der theoretischen Literatur (vgl. Ludewig/Lichter 2010, 370f.). Da alle Befragten ein ausgeprägtes Wissen über Usability haben, sehen sie in den Anforderungsmustern keinen Wissenszuwachs, sondern vielmehr eine standardisierte Wissensgrundlage. Diese könne durchaus dabei helfen, Designs zu verbessern, zu vervollständigen und auf Usability-Probleme aufmerksam zu machen.

Der Umstand, dass alle Befragten Usability in der Entwicklung berücksichtigen und gleichzeitig keine Anforderungsmuster verwenden, hat Einfluss auf die allgemeine Akzeptanz der Muster. So werden die Muster zwar inhaltlich anerkannt, ein

Verwenden der Muster im Arbeitsalltag kommt für die Befragten in der jetzigen Form nicht in Frage.

Es ist davon auszugehen, dass Usability aufgrund seiner hohen Bedeutung bei allen erfolgreichen Software-Entwicklern explizit berücksichtigt wird. Obwohl die Kriterien der Auffindbarkeit, Verständlichkeit und Hilfestellung ausreichend positiv bewertet wurden, finden die erarbeiteten Usability-Anforderungsmuster bei den Befragten keine vollständige Akzeptanz, da ihr Inhalt bereits in anderer Form berücksichtigt wird. Dabei fällt auf, dass offenbar keine vergleichbare Form der Usability-Dokumentation zum Einsatz kommt: bei den befragten Unternehmen existiert zwar das Wissen über Gestaltungstechniken zur Schaffung von Usability, jedoch werden im Vorhinein keine Usability-Anforderungen an die Projekte gestellt und dokumentiert. Dies führt zu der Frage, wie Usability als Ziel der Softwareentwicklung überprüft werden kann, wenn gleichzeitig keine Usability-Tests stattfinden.

Die Ausgangsfrage war, ob die Usability-Anforderungsmuster in der Praxis gut anwendbar sind. Die Befragten begrüßen die Verwendung eines Usability-Leitfadens, wünschen sich diesen allerdings nicht in Form von Anforderungsmustern. Um die Beantwortung der Forschungsfrage fortzusetzen, wären weitere Befragungen von Unternehmen notwendig, die nicht nur mit Anforderungen, sondern auch mit Anforderungsmustern arbeiten. Alternativ kann erfragt werden, in welcher Form Entwickler in der Praxis Usability-Empfehlungen dokumentieren und anwenden. Für Unternehmen, die keine Anforderungsmuster verwenden, empfiehlt sich möglicherweise die Verwendung einer Usability-Checkliste, wie sie A vorschlug.

6 Anwendung der Usability-Anforderungsmuster an einem Fallbeispiel

Nachdem zuvor Anforderungsmuster für Usability zusammengestellt und diskutiert wurden, sollen die Muster im Folgenden angewendet werden, um Anforderungen für ein Fallbeispiel abzuleiten. Ziel ist es, die Anwendbarkeit der Muster darzustellen und zu prüfen, ob auf Grundlage der Muster gut benutzbare Bedienoberflächen gestaltet werden können. Dazu werden zunächst sinnvolle Usability-Anforderungen für den Prototyen einer Smartphone-Applikation zusammengestellt und anschließend Gestaltungsvorschläge erarbeitet, die diese Anforderungen berücksichtigen.

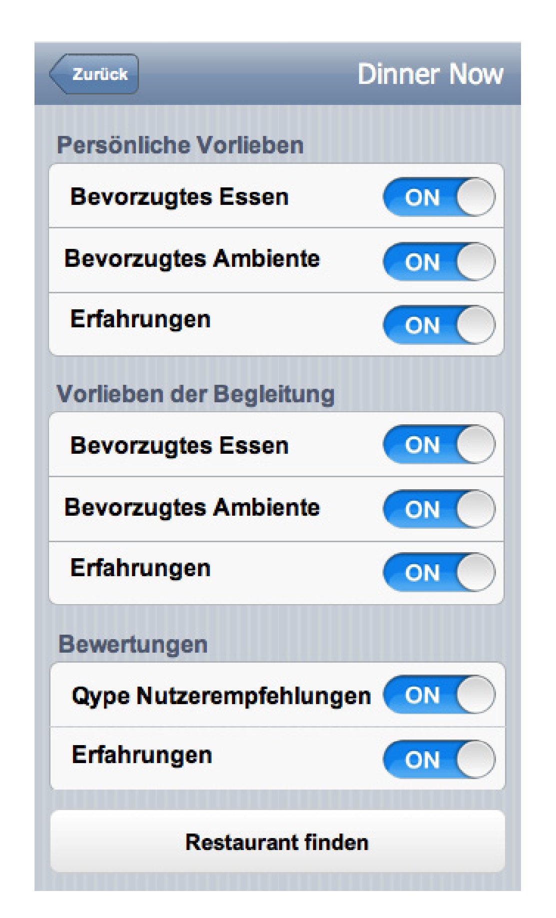
6.1 Vorstellung des Fallbeispiels „Dinner Now“

Dinner Now ist eine Smartphone-Applikation, kurz App, für das Betriebssystem iOS. Die Beschreibung der App lautet wie folgt:

> „Dinner Now ermöglicht dem Nutzer, das passende Restaurant für sich und seine Begleiter, basierend auf den jeweiligen Vorlieben und dem aktuellen Ort, zu finden. Der Nutzer hat hierbei die Möglichkeit, sowohl seine persönlichen Vorlieben, als auch die Vorlieben seiner Begleitung bezüglich der Art des Essens (Nationalität), des Ambientes und Vorerfahrungen mit in die Empfehlung einzubeziehen. Ebenso lassen sich Nutzermeinungen aus Webportalen für die Generierung der Empfehlung berücksichtigen. Vorlieben von Nutzern und Begleitern sowie die Nutzermeinungen werden laut Spezifikation der Anwendung aus den Profilen der Nutzer in sozialen Netzwerken ausgelesen. Hat der Nutzer die Einstellungen getätigt, wird ihm das am besten passende Restaurant, inklusive einiger Details, angezeigt. Zusätzlich hat er die Möglichkeit, im Restaurant anzurufen oder sich die Route zum Restaurant anzeigen zu lassen. Ist er mit der von Dinner Now vorgeschlagenen Empfehlung nicht zufrieden, kann sich der Benutzer einen neuen Vorschlag generieren lassen. Zusätzlich hat er zu jeder Zeit die Möglichkeit, wieder von vorne zu beginnen und zum Start der Anwendung zurückzukehren.“ (Söllner et al. 2012, 123f.).

Die App befindet sich im Prototypen-Stadium. Derzeit existieren 3 Beispielbildschirme, die die mögliche Benutzeroberfläche der App zeigen. Der erste

Beispielbildschirm zeigt ein Einstellungsmenü, in dem der Nutzer entscheidet, welche Faktoren bei der Suche berücksichtigt werden sollen: die eigenen Vorlieben, die Vorlieben der Begleitung und/oder Bewertungen aus anderen Netzwerken. Der Prototyp geht davon aus, dass zu diesem Zeitpunkt bereits entsprechende Vorlieben eingegeben wurden. Da es heute noch nicht möglich ist, standardisierte Essensvorlieben aus den Profilangaben sozialer Netzwerke auszulesen, wird diese Funktion sowohl in der folgenden Anforderungsspezifikation als auch in den Gestaltungsvorschlägen nicht ausgestaltet. Gleiches gilt für die Erfahrungen, die bei der Restaurantsuche mit einbezogen werden sollen. Da diese Funktion bisher nicht ausreichend definiert wurde, wird sie im neuen Entwurf nicht bearbeitet.



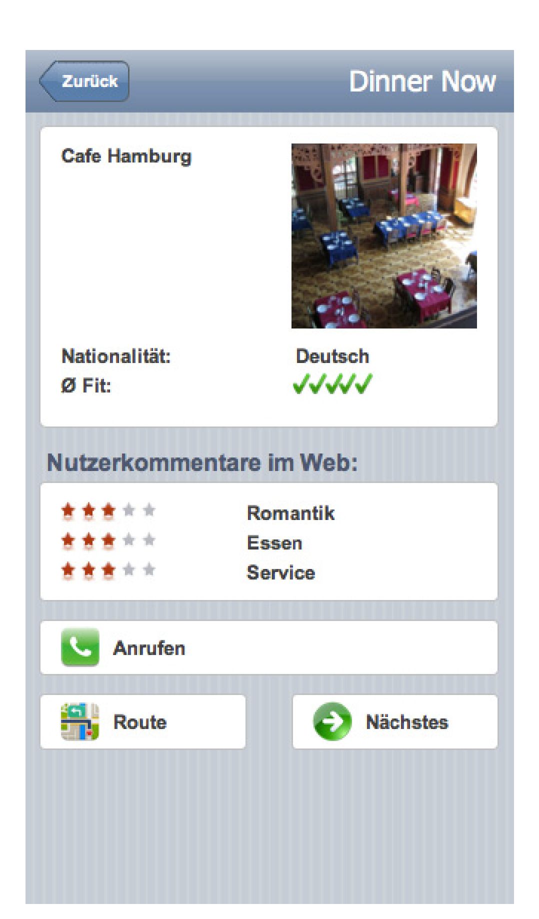


Abbildung 2: Original-Bildschirmausschnitt 1 – Dinner Now Sucheinstellungen und Vorschlag
Quelle: Söllner et al. 2012, 125

Beim Antippen von „Restaurant finden“ wird ein passender Vorschlag angezeigt, zu sehen auf Abbildung 2. Auf der Vorschlagsseite sind Fotos, die Nationalität des Essens und Nutzerbewertungen zu sehen. Sollte der Vorschlag dem Nutzer nicht gefallen, kann er mit Antippen von „Nächstes“ den nächsten Vorschlag aufrufen.

Sofern der Vorschlag gefällt, kann mit "Route" der Weg auf Google Maps, einem Online- und GPS-gestütztem Kartendienst, angezeigt werden. Wie auf der folgenden Abbildung zu sehen ist, werden neben der Route auch die Entfernung und die ungefähre Gehzeit angegeben. Links oben ist jederzeit ein „Zurück"-Button zu sehen, der den Nutzer beim Antippen zu dem jeweils letzten Dialog zurück bringt.

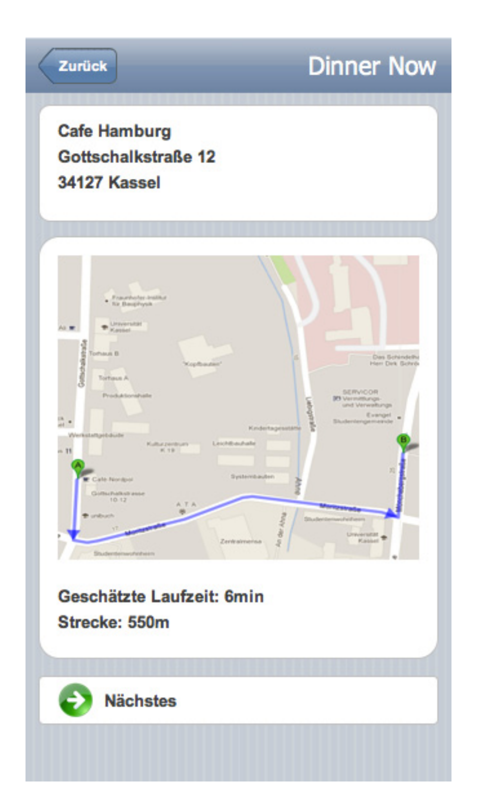


Abbildung 3: Original-Bildschirmausschnitt 2 – Route zu Restaurant
Quelle: Söllner et al. 2012, 125

Der Unterschied zu vielen bereits existierenden Restaurant-Finder-Applikationen ist, dass mehrere Vorlieben gleichzeitig berücksichtigt werden können, um die Entscheidung für ein Restaurant innerhalb einer Gruppe zu erleichtern. Außerdem werden nach Betätigen der Suche direkt einzelne Vorschläge in einer Detailansicht gezeigt, während andere Restaurant-Finder zuerst Ergebnislisten anzeigen.

Da die App darauf abzielt, alle möglichen Essensvorlieben abzudecken, wird eine sehr breite Zielgruppe anvisiert. Das macht wiederum Usability zu einem entscheidenden Erfolgsfaktor: die App soll sowohl einen Informations- als auch einen Bequemlichkeitsnutzen bieten. Damit eine solche App Erfolg hat, muss ihre Benutzung einfach und befriedigend sein. Die Anwendung der Usability-Anforderungsmuster ist damit gerechtfertigt.

6.2 Erstellung von Usability-Anforderungen

Mit Hilfe des erarbeiteten Katalogs aus Usability-Anforderungsmustern sollen nun Usability-Anforderungen an Dinner Now gestellt werden. Dazu werden die einzelnen Muster der Reihe nach auf Eignung und Anwendbarkeit hin geprüft. Für den genauen Wortlaut und die Detailangaben der einzelnen Muster wird auf Anhang A verwiesen. Die mit den Mustern erstellten Anforderungen erhalten Nummern mit dem Kürzel DN für Dinner Now.

Bevor die Muster angewendet werden können, sollten einige häufig auftauchende Begriffe aus der Prozesswortliste auf die App angewendet werden. Hierbei handelt es sich v.a. um Wörter, die als Parameter in den Vorlagentexten der Muster genannt werden und deren mögliche Ausprägungen bekannt sein sollten:

Aufgaben:
- Finden eines passenden Restaurants
- Auffinden eines gewählten Restaurants

Funktionen:
- Vorlieben eingeben
- Restaurant-Suche durchführen
- Ergebnisse anzeigen
- Route anzeigen

Aktionen:
- Antippen von Bedienelementen
- Eingeben von Zeichen

Inhalte:
- Restaurant-Vorschläge
- Routen

Einstellungen:
- Berücksichtigte Suchfaktoren
- Vorlieben des Nutzers
- Vorlieben der Begleitung

Begonnen wird mit Anforderungsmustern zum Bereich der Kontrolle und der Einschränkungen. Die erste mögliche Anforderung besagt, dass der Nutzer auswählt, welche Funktionen die Applikation durchführt und welche nicht. Da sich Nutzer mit Dinner Now zielgerichtet über Restaurants informieren wollen, sie also eine konkrete Aufgabe verfolgen, sollte die Anforderung angewendet werden:

DN-KO-01: Explizite Aktionskontrolle
Die Applikation soll nur Funktionen ausführen, die der Nutzer mit seinen Aktionen ausgelöst hat.

Um die App steuern zu können, sollte eine Form der Navigation vorliegen, die dem Nutzer die verschiedenen Funktionen der App zugänglich macht:

DN-KO-02: Steuerbarkeit durch Vorliegen einer Navigation
Die Applikation soll dem Nutzer ermöglichen, sich innerhalb der Dialoge vor- und zurückzubewegen.

Wurden Funktionen erst einmal ausgewählt, sollte der Nutzer deren Durchführung beherrschen können. Dazu sollen Funktionen pausiert und abgebrochen werden können. Das Pausieren von Funktionen bietet sich an, wenn Funktionen längere Zeit dauern, z. B. beim Abspielen einer Musikdatei oder bei einer umfangreichen Berechnung. Keine der Funktionen von Dinner Now dauert – eine ausreichende Datenverbindung vorausgesetzt – längere Zeit, sodass ein mögliches Pausieren nicht als Anforderung mit aufgenommen wird. Gerade aber aufgrund möglicher Verzögerungen im Datenverkehr mit dem Funknetz oder Satelliten kann die Ausführung einer der Funktionen mehr als ein paar Sekunden dauern. Es empfiehlt sich daher eine Anforderung zum Unterbrechen von Funktionen.

DN-KO-03: Funktionen pausieren und abbrechen
Während die Applikation eine Funktion ausführt, soll der Nutzer die Möglichkeit haben, die Funktion abzubrechen.

KO-04 und KO-05 beschäftigen sich mit Einstellungsmenüs und Vor-Einstellungen. Nutzer sollen wissen, wenn Vor-Einstellungen existieren und sollen diese verändern können. Der Hauptzweck von Dinner Now ist eine Suche nach bestimmten Kriterien. Einstellungsmenüs spielen daher eine wichtige Rolle, weshalb beide Anforderungsmuster angewendet werden:

DN-KO-04: Informationen zu Vor-Einstellungen
Wenn die Applikation in den Sucheinstellungen selbstständig Einstellungen vornimmt, soll die Applikation den Nutzer darauf hinweisen.

DN-KO-05: Vor-Einstellungen abspeichern
Wenn die Sucheinstellungen auszufüllen sind, soll die Applikation dem Nutzer die Möglichkeit bieten, Einstellungen abzuspeichern und wiederzuverwenden.

Das Muster KO-06 schlägt vor, bei komplexen Aufgaben zwei Wege zur Aufgabenbewältigung anzubieten; einen einfachen Weg mit wenigen Optionen und einen Weg mit allen Optionen für erfahrene Nutzer. Der einfache Weg soll Erstnutzern den Einstieg erleichtern (vgl. Lidwell et al. 2003, S. 64). Es kann im Vorhinein davon ausgegangen werden, dass die Aufgabe der Restaurantfindung sehr simpel gestaltet werden kann. Da die Suche jedoch Einstellungen erlaubt, kann durch das Anbieten einer Suche mit Vor-Einstellungen durchaus ein Nutzen für Einsteiger oder bequeme Nutzer entstehen.

DN-KO-06: Zwei Wege zur Aufgabenbewältigung
Wenn die Restaurant-Suche Einstellungen erlaubt, soll die Applikation die Funktion einmal mit Vor-Einstellungen und einmal mit allen Einstellungen anbieten.

Das letzte Anforderungsmuster zum Bereich Kontrolle schlägt vor, dem Nutzer nur Funktionen anzubieten, die aktuell verfügbar sind. Da Dinner Now nur vier Funktionen anbietet, die stark miteinander verwoben sind, lassen sich für dieses Muster keine Anwendungsbereiche finden. Ausnahme sind Funktionen, die wegen fehlender Funk- oder Satellitenanbindung nicht ausgeführt werden können. Besteht beispielsweise keine Internetverbindung, kann die App keine Restaurantvorschläge anbieten. In diesem Fall sollte die Suchfunktion der App jedoch nicht verborgen werden, sondern beim Auslösen eine hilfreiche Fehlermeldung anbieten, weshalb hier auf DN-FE-02: *Fehler-Benachrichtigungen* verwiesen wird.

Die folgenden Mustern behandeln den Bereich der Effizienz. Ziel ist es hier, die App geradlinig zu gestalten, sodass sie schnell und ohne große kognitive Belastung genutzt werden kann. Dinner Now soll zum spontanen Finden von Restaurants Nutzergruppe verwendet werden. Darüber hinaus wird davon ausgegangen, dass der Nutzer während der Anwendung nicht alleine ist und sich mit seiner Begleitung austauscht. Der Nutzungskontext birgt daher viele Ablenkungen. Dies begründet die Aufstellung vieler Anforderungen aus dem Bereich Effizienz, die das Benutzen von Dinner Now mit geringer oder kurzer Aufmerksamkeit sicherstellen. EF-01 hat zum Ziel, dargestellte Inhalte der App hervorzuheben. Die von Dinner Now dargestellten Inhalte sind in

erster Linie die Restaurant-Vorschläge und die dargestellten Routen. Die Parameter für den Inhalt sind damit Vorschläge und Routen. Es ergeben sich die folgenden Anforderungen:

DN-EF-01: Inhalt herausstellen
Zeigt die Applikation Restaurant-Vorschläge an, sollen die Bedienelemente nicht von diesen Vorschlägen ablenken.

Zeigt die Applikation Routen an, sollen die Bedienelemente nicht von diesen Vorschlägen ablenken.

Es wird weiterhin versucht, den Bedienaufwand des Nutzers zu minimieren, um die schnelle, beiläufige Nutzung der Applikation zu ermöglichen. EF-02 und EF-03 werden in standardisierter Form angewendet:

DN-EF-02: Aufwand des Nutzers minimieren
Der Nutzer soll nur Aktionen ausführen müssen, die seine Entscheidung erfordern.

DN-EF-03: Körperlichen Einsatz minimieren
Das Durchführen von Aktionen soll mit dem geringstmöglichen körperlichen Einsatz erfolgen.

Das nächste Anforderungsmuster versucht sicherzustellen, dass Text- und Bildanweisungen verständlich sind. Sofern solche Anweisungen bei Dinner Now auftauchen, sollten diese Anforderungen erfüllt sein, damit die App ohne langes Nachdenken ausprobiert werden kann.

DN-EF-04: Verständliche Anweisungen
Wenn der Nutzer eine Aktion durchführen soll, soll die Applikation Bildanweisungen mit unmissverständlichen Bildern und Symbolen verwenden.

Wenn der Nutzer eine Aktion durchführen soll, soll die Applikation Textanweisungen mit kurzen, klaren Sätzen verwenden.

EF-05 empfiehlt, komplexe Aufgaben in Teilschritte aufzuteilen. Eine Aufgabe ist als komplex zu betrachten, wenn Erstnutzer bei der Durchführung überfordert sein

könnten (vgl. Hix/Hartson 1993, 35). Die App möchte dem Nutzer helfen, ein passendes Restaurant zu finden. Um diese Aufgabe durchzuführen, müssen Suchfaktoren festgelegt, eine Suche durchgeführt und passende Vorschläge betrachtet werden. Es empfiehlt sich, die Aufgabenerfüllung aufzuteilen und das Anforderungsmuster anzuwenden:

> **DN-EF-05: Aufteilen komplexer Aufgaben**
> Die für das Finden eines Restaurants notwendigen Aktionen sollen in Teilschritte unterteilt werden.

Damit die App übersichtlich bleibt, empfiehlt EF-06, dass nur Informationen angezeigt werden, die für die aktuelle Aufgabe notwendig sind. Da mit einem Smartphone-Bildschirm nur eine kleine Darstellungsfläche zur Verfügung steht und die App beiläufig genutzt werden soll, wird auch diese Anforderung aufgenommen.

> **DN-EF-06: Übersichtlichkeit**
> Die Applikation soll nur Informationen anzeigen, die zur aktuell ausgewählten Funktion gehören.

Um die kognitive Belastung des Nutzers weiter zu verringern, soll er sich keine Informationen von einem Schritt bis zum nächsten merken müssen. Dies gilt für jede Funktion, die sich in mehrere Schritte oder Aktionen aufteilt. Parameter ist in diesem Fall das Durchführen der Restaurant-Suche.

> **DN-EF-07: Kein Merken-Müssen von Informationen**
> Wenn die Suche nach Restaurants mit mehreren Aktionen durchgeführt wird, soll die Applikation dem Nutzer stets alle notwendigen Informationen anzeigen.

Die folgenden Anforderungsmuster befassen sich mit dem Bereich der Personalisierung. Muster IN-01 schlägt vor, dass Nutzer aktiv Einstellungen an den Funktionen der Applikation vornehmen können sollen. Das stellt bezogen auf Dinner Now eine Kern-Anforderung dar, da hier eine Restaurant-Suche nach ausgewählten Vorlieben stattfindet.

> **DN-IN-01: Aktive Personalisierung anbieten**
> Die Applikation soll dem Nutzer erlauben, Einstellungen an der Restaurant-Suche vorzunehmen.

Das Muster IN-02 enthält die Anforderung, den Nutzer nicht verwendete Optionen ausblenden zu lassen. Da Dinner Now eine von vornherein sehr geradlinige Applikation mit nur einer Hauptfunktion ist und nur wenige Optionen anbietet, ist ein Aufnehmen dieser Anforderung nicht notwendig. IN-03 beschreibt eine Anforderung zum gleichzeitigen Ausführen mehrerer Aktionen durch einen einzigen Befehl. Dies wird üblicherweise über Abkürzungen oder Tastenkürzel gelöst. Da Dinner Now bereits sehr kurze Wege zur Aufgabenlösung enthält, erscheint ein Anbieten weiterer Abkürzungen als nicht notwendig.

In den kommenden Anforderungsmustern wird der Umgang mit Fehlern behandelt. Die erste mögliche Anforderung ist das Bestätigen kritischer Aktionen aus FE-01, bei dem Nutzer bei Auslösen einer kritischen Funktion diese ein zweites Mal bestätigen müssen. Kritische Funktionen sind solche, deren Folgen nur schwer oder gar nicht rückgängig gemacht werden können (vgl. Lidwell/Holden/Butler 2003, 54). Dinner Now bietet ein bequemes Suchen von Informationen. Kritische Funktionen im oben genannten Sinn tauchen im Grundkonzept der Applikation nicht auf. Lediglich das Verbinden der App mit sozialen Netzwerken kann – je nach Netzwerk – irreversible Folgen haben und wäre daher als kritisch einzustufen. Diese Funktion wird hier jedoch ausgeblendet.

FE-02 schlägt das Anbieten von Benachrichtigungen vor, wenn der Nutzer Fehler macht. Benutzungsfehler sind bei der Verwendung von Dinner Now durchaus denkbar, z. B. wenn der Anwender Suchkriterien für die Restaurant-Suche aktiviert, diese Kriterien jedoch noch nicht mit Inhalt hinterlegt wurden. Es werden deshalb Anforderungen zu den Fehler-Benachrichtigungen aufgenommen:

> **DN-FE-02: Fehler-Benachrichtigungen**
> Macht der Nutzer einen Fehler, soll die Applikation ihn unmittelbar darauf hinweisen.
>
> Zeigt die Applikation eine Fehlerbenachrichtigung, soll diese die Art des Fehlers, seine Position und einen Korrekturhinweis anbieten.

Unabhängig von gemachten Fehlern sollten Anwender stets die Möglichkeit haben, Aktionen und Eingaben rückgängig zu machen. Diese Grundregel sollte auch bei Dinner Now Anwendung finden:

DN-FE-03: Aktionen rückgängig machen
Der Nutzer sollte die Möglichkeit haben, Aktionen und Eingaben unmittelbar rückgängig zu machen.

Wenn Nutzer in einem Dialog falsche Eingaben machen, ist es bequem, die falschen Eingaben korrigieren zu können, ohne die richtigen Eingaben neu eingeben zu müssen. Falsch-Eingaben können beim Einstellen der Essens-Vorlieben entstehen, weshalb diese Anforderung sinnvoll ist.

DN-FE-04: Isoliertes Eingeben von falschen Eingaben
Macht der Nutzer Fehler, soll er falsche Eingaben unmittelbar korrigieren können, ohne die richtigen Eingaben neu eingeben zu müssen.

Wenn Eingabeformulare existieren, sollten Nutzer ihre Eingaben vor Auslösen einer Funktion überprüfen und verändern können. Dies gilt v.a. für kritische Funktionen, bei denen vielfältige Eingaben zu tätigen sind. Da bei Dinner Now keine kritischen Funktionen vorliegen, muss Anforderungsmuster FE-05 nicht explizit berücksichtigt werden. Wenn Nutzer mit einer Applikation Inhalte erstellen, können diese bei einem Überschreiben automatisch gespeichert werden, um ein eventuelles Rückgängigmachen zu ermöglichen. Da die Nutzer von Dinner Now mit keiner Funktion selbst Inhalte schaffen, muss dieses automatische Speichern ebenfalls nicht als Anforderung aufgenommen werden.

Die nachfolgenden Muster befassen sich mit dem Schaffen von Startpunkten für Applikationen und deren Funktionen. Die erste mögliche Anforderung fordert das schlichte Vorliegen eines solchen Startpunktes. Dieser fehlt bei den ursprünglichen Entwürfen für Dinner Now; die Anforderung wird deshalb aufgenommen.

DN-EP-01: Startpunkt anbieten
Die Applikation soll einen offensichtlichen Startpunkt haben.

Ein Startpunkt sollte stets erreichbar sein, keine wichtigen Funktionen oder Informationen verbergen und stattdessen einen Überblick über die Funktionen der Applikation bieten. Um den Startpunkt von Dinner Now zu gestalten, kommen alle diese Anforderungen zum Einsatz.

DN-EP-02: Jederzeitiges Zurückkehren zum Startpunkt
Der Nutzer soll jederzeit die Möglichkeit haben, zum Startbildschirm zurückzukehren.

DN-EP-03: Keine Barrieren im Startpunkt
Der Startpunkt der Applikation soll das unmittelbare Verwenden ihrer Funktionen erlauben.

DN-EP-04: Überblick-Funktion des Startpunkts
Der Startpunkt soll einen Überblick über die Funktionen der Applikation enthalten.

Auf dem Startpunkt aufbauend müssen sich Nutzer auch auf allen nachfolgenden Seiten der Applikation zurechtfinden können. Da Dinner Now's Benutzung einfach sein soll und Verwirrung ausgeschlossen sein muss, empfehlen sich die zwei Anforderungen für den Bereich der Wegfindung:

DN-WE-01: Orientierung anbieten
Die Applikation soll dem Nutzer zu jedem Zeitpunkt anzeigen, an welcher Stelle der Navigation er sich befindet.

DN-WE-02: Aussagekräftige Bedienelemente
Die Navigation soll aussagekräftige und klar benannte Bedienelemente haben.

Im Folgenden werden Anforderungen für die richtige Anordnung von Informationen aufgestellt. Dazu sollen Informationen in ihrer Darstellung hierarchisiert und gruppiert werden. Dinner Now bietet durchaus unterschiedliche Arten von Informationen an. Es ist zur Berücksichtigung der Anforderungen festzustellen, welche Informationen zusammengehörig sind und welche wichtiger sein können als andere.

DN-AI-01: Wichtige Informationen hervorheben
Die Applikation soll wichtige Informationen gegenüber weniger wichtigen Informationen hervorgehoben darstellen.

DN-AI-02: Informationen gruppieren
Die Applikation soll Informationen in der Darstellung gruppieren.

Damit die zuvor genannten Steuerungsmöglichkeiten stets zugänglich sind, sollte die Applikation stets Zugriff auf die Navigation erlauben. So kann ein Nutzer die Verwendung so lange fortsetzen, bis er ein passendes Restaurant gefunden hat.

DN-AI-03: Ständig erreichbare Navigation
Die Navigation der Applikation soll immer erreichbar sein.

Damit eine Applikation als etwas Zusammenhängendes wahrgenommen wird, ist eine konsistente Gestaltung notwendig. Da Dinner Now mehrere Funktionen hat, empfiehlt sich die Anwendung von Anforderungsmustern zum Bereich Konsistenz.

DN-CO-01: Alle Teile einer Applikation ähnlich gestalten
Teilt sich die Applikation in verschiedene Bereiche auf, sollen die Bereiche identisch formatiert sein.

Werden Funktionen und Informationen an mehreren Stellen innerhalb der Applikation angeboten, sollten sie überall ähnlich gestaltet sein, um Nutzer nicht zu verwirren. Da Dinner Now eine überschaubare Funktionalität aufweist, werden sowohl Informationen als auch einzelne Funktionen immer nur an einer Stelle angeboten. Deshalb werden die Muster CO-02: *Konsistent gestaltete Funktionen* und CO-03: *Konsistent gestaltete Informationen* nicht angewendet. In Verbindung mit dem Funktionieren der Applikation auf jedem Endgerät aus DN-IN-01 wird im Rahmen der Konsistenz gefordert, eine Applikation auf jedem Endgerät gleich aussehen zu lassen.

DN-CO-04: Konsistente Darstellung auf verschiedenen Endgeräten
Wird die Applikation von unterschiedlichen Endgeräten aus genutzt, soll die Applikation auf jedem Endgerät identisch dargestellt werden.

Eine Software sollte ihrem Anwender immer mitteilen, ob sie etwas bearbeitet. Zunächst sollte sie stets den aktuellen Systemstatus anzeigen; SI-01 wird als standardisierte Anforderung übernommen.

DN-SI-01: Systemstatus anzeigen
Die Applikation soll den Nutzer über den aktuellen Status des aktuell ausgewählten Bereiches informieren.

Anforderungen zum Bereich Rückmeldung stellen sicher, dass Nutzer bei Verwendung der Applikation stets ein ausreichendes Feedback bekommen. Rückmeldungen geben

Nutzern ein Gefühl von Sicherheit sowie Kontrolle und machen die Benutzung einer Software angenehm. Entsprechende Anforderungen sollten daher auch bei Dinner Now angewendet werden, damit Nutzer wissen, dass ihre Aktionen etwas bewirken. Bedenkt man die Nutzungssituation, kann von Nutzern nur wenig Geduld erwartet werden. Deshalb werden folgende Anforderungen aufgestellt:

DN-RM-01: Rückmeldung bei Aktionen
Wenn der Nutzer eine Aktion durchführt, soll die Applikation ihn unmittelbar und wahrnehmbar über die ausgelöste Bearbeitung informieren.

DN-RM-02: Fortschrittsanzeige bei langer Bearbeitung
Wenn der Nutzer eine Funktion auslöst, deren Bearbeitung sehr lange dauert, soll die Applikation ihn unmittelbar und wahrnehmbar über die Bearbeitung informieren.

DN-RM-03: Sofortiges Anzeigen von Eingaben
Wenn der Nutzer Eingaben in einem Eingabefeld macht, soll die Applikation unmittelbar anzeigen, ob und wie diese Eingaben im entsprechenden Eingabefeld erfasst wurden.

DN-RM-04: Erfolgreiche Bearbeitung anzeigen
Wenn der Nutzer eine Funktion ausgelöst hat, soll die Applikation ihn nach der Bearbeitung darüber informieren, ob die Funktion erfolgreich ausgeführt wurde oder nicht.

Bei allen Gestaltungsoptionen sind etablierte Vorgehensweisen, sog. „Best Practices“ zu beachten. Dies erleichtert Nutzern das Erlernen der Applikation und sollte auch bei Dinner Now berücksichtigt werden. Da sowohl die Benutzung von Smartphones, als auch die Bedienoberfläche von iOS vielen Menschen bekannt ist, ergeben sich daraus Gestaltungskonventionen. Hinzu kommt, dass auch Sucheinstellungen oder sonstige Einstellungen aus vielen mobilen Anwendungen bekannt sind. Mit den gängigen Darstellungsmethoden sollte daher nicht gebrochen werden.

DN-EK-01: Beachten von Gestaltungs-Best Practices
Wenn für die Gestaltung einer Funktion Best Practices existieren, soll die Applikation diese Best Practices einhalten.

Um die Lesbarkeit der Bildschirmtexte zu gewährleisten, werden beide Anforderungen zu diesem Bereich aufgenommen:

DN-LE-01: Gut lesbarer Text
Die Applikation soll für Fließtext einen Schriftgrad zwischen 9 und 12 Pixel verwenden.

Die Applikation soll Schrift mit einem hohen, aber nicht absoluten Farbkontrast gegenüber dem Hintergrund darstellen.

Die Applikation soll Schriften verwenden, die auf allen Betriebssystemen verfügbar sind.

DN-LE-02: Verständlicher Text
Zeigt die Applikation Informationen in Form von Text an, soll die Applikation Begriffe verwenden, die die Nutzer verstehen.

Schlussendlich wird die Benutzbarkeit einer Applikation dadurch erhöht, dass eine Hilfestellung vorliegt, die Nutzern die richtige Benutzung erklärt. Es ist möglich, komplexere Funktionen mit begleitenden Hinweisen zu versehen, um den Nutzer zu führen. Da Dinner Now eine sehr eindeutige Funktionalität hat, ist ein solches begleitendes Unterstützen nicht notwendig. Fehlt die begleitende Hilfestellung, muss diese auch nicht übersprungen werden. Die Muster HI-02: *Komplexe Funktionen mit Hinweisen* begleiten und HI-03: *Hilfestellung überspringen* kommen nicht zur Anwendung.

DN-HI-01: Vorliegen einer Hilfestellung
Die Applikation soll eine Hilfestellung anbieten.

Mit den angewendeten Mustern ergibt sich die folgende Anforderungsspezifikation für das Berücksichtigen von Usability bei Dinner Now. Von den 47 Usability-Anforderungsmustern kamen 37 zur Anwendung.

U-01	DN-KO-01	Explizite Aktionskontrolle	Die Applikation soll nur Funktionen ausführen, die der Nutzer mit seinen Aktionen ausgelöst hat.
U-02	DN-KO-02	Steuerbarkeit durch Vorliegen einer Navigation	Die Applikation soll dem Nutzer ermöglichen, sich innerhalb der Dialoge vor- und zurückzubewegen.
U-08	DN-KO-03	Funktionen abbrechen	Während die Applikation eine Funktion ausführt, soll der Nutzer die Möglichkeit haben, die Funktion abzubrechen.
U-04	DN-KO-04	Informationen zu Vor-Einstellungen	Wenn die Applikation in den Sucheinstellungen selbstständig Einstellungen vornimmt, soll die Applikation den Nutzer darauf hinweisen.
U-05	DN-KO-05	Vor-Einstellungen abspeichern	Wenn die Sucheinstellungen auszufüllen sind, soll die Applikation dem Nutzer die Möglichkeit bieten, Einstellungen abzuspeichern und wiederzuverwenden.
U-06	DN-KO-06	Zwei Wege zur Aufgabenbewältigung	Wenn die Suchfunktion Einstellungen erlaubt, soll die Applikation die Funktion einmal mit Vor-Einstellungen und einmal mit allen Einstellungen anbieten.
U-07	DN-EF-01	Inhalt herausstellen	Zeigt die Applikation Restaurant-Vorschläge an, sollen die Bedienelemente nicht von diesen Vorschlägen ablenken. Zeigt die Applikation Routen an, sollen die Bedienelemente nicht von diesen Vorschlägen ablenken.
U-08	DN-EF-02	Aufwand des Nutzers minimieren	Der Nutzer soll nur Aktionen ausführen müssen, die seine Entscheidung erfordern.
U-09	DN-EF-03	Körperlichen Einsatz minimieren	Das Durchführen von Aktionen soll mit dem geringstmöglichen körperlichen Einsatz erfolgen.
U-10	DN-EF-04	Verständliche Anweisungen	Wenn der Nutzer eine Aktion durchführen soll, soll die Applikation Textanweisungen mit kurzen, klaren Sätzen verwenden.
U-11	DN-EF-05	Aufteilen komplexer Aufgaben	Die für das Finden eines Restaurants notwendigen Aktionen sollen in Teilschritte unterteilt werden.
U-12	DN-EF-06	Übersichtlichkeit	Die Applikation soll nur Informationen anzeigen, die zur aktuell ausgewählten Funktion gehören.
U-13	DN-EF-07	Kein Merken-Müssen von Informationen	Wenn die Suche nach Restaurants mit mehreren Aktionen durchgeführt wird, soll die Applikation dem Nutzer stets alle notwendigen Informationen anzeigen.
U-14	DN-IN-01	Aktive Personalisierung anbieten	Die Applikation soll dem Nutzer erlauben, Einstellungen an der Restaurant-Suche vorzunehmen.
U-15	DN-FE-02	Fehler-Benachrichtigungen	Macht der Nutzer einen Fehler, soll die Applikation ihn unmittelbar darauf hinweisen. Zeigt die Applikation eine Fehlerbenachrichtigung, soll diese die Art des Fehlers, seine Position und einen Korrekturhinweis anbieten.
U-16	DN-FE-03	Aktionen rückgängig machen	Der Nutzer sollte die Möglichkeit haben, Aktionen und Eingaben unmittelbar rückgängig zu machen.
U-17	DN-FE-04	Isoliertes Eingeben von falschen Eingaben	Macht der Nutzer Fehler, soll er falsche Eingaben unmittelbar korrigieren können, ohne die richtigen Eingaben neu eingeben zu müssen.
U-18	DN-EP-01	Startpunkt anbieten	Die Applikation soll einen offensichtlichen Startpunkt haben.
U-19	DN-EP-02	Jederzeitiges Zurückkehren zum Startpunkt	Der Nutzer soll jederzeit die Möglichkeit haben, zum Startpunkt zurückzukehren.
U-20	DN-EP-03	Keine Barrieren im Startpunkt	Der Startpunkt der Applikation soll das unmittelbare Verwenden ihrer Funktionen erlauben.
U-21	DN-EP-04	Überblick-Funktion des Startpunkts	Der Startpunkt soll einen Überblick über die Funktionen der Applikation enthalten.
U-22	DN-WE-01	Orientierung anbieten	Die Applikation soll dem Nutzer zu jedem Zeitpunkt anzeigen, an welcher Stelle der Navigation er sich befindet.

U-23	DN-WE-02	Aussagekräftige Bedienelemente	Die Navigation soll aussagekräftige und klar benannte Bedienelemente haben.
U-24	DN-AI-01	Wichtige Informationen hervorheben	Die Applikation soll wichtige Informationen gegenüber weniger wichtigen Informationen hervorgehoben darstellen.
U-25	DN-AI-02	Informationen gruppieren	Die Applikation soll Informationen in der Darstellung gruppieren.
U-26	DN-AI-03	Ständig erreichbare Navigation	Die Navigation der Applikation soll immer erreichbar sein.
U-27	DN-CO-01	Alle Teile einer Applikation ähnlich gestalten	Teilt sich die Applikation in verschiedene Bereiche auf, sollen die Bereiche identisch formatiert sein
U-28	DN-CO-04	Konsistente Darstellung auf verschiedenen Endgeräten	Wird die Applikation von unterschiedlichen Endgeräten aus genutzt, soll die Applikation auf jedem Endgerät identisch dargestellt werden.
U-29	DN-SI-01	Systemstatus anzeigen	Die Applikation soll den Nutzer über den aktuellen Status des aktuell ausgewählten Bereiches informieren.
U-30	DN-RM-01	Rückmeldung bei Aktionen	Wenn der Nutzer eine Aktion durchführt, soll die Applikation ihn unmittelbar und wahrnehmbar über die ausgelöste Bearbeitung informieren.
U-31	DN-RM-02	Fortschrittsanzeige bei langer Bearbeitung	Wenn der Nutzer eine Funktion auslöst, deren Bearbeitung sehr lange dauert, soll die Applikation ihn unmittelbar und wahrnehmbar über die Bearbeitung informieren.
U-32	DN-RM-03	Sofortiges Anzeigen von Eingaben	Wenn der Nutzer Eingaben in einem Eingabefeld macht, soll die Applikation unmittelbar anzeigen, ob und wie diese Eingaben im entsprechenden Eingabefeld erfasst wurden.
U-33	DN-RM-04	Erfolgreiche Bearbeitung anzeigen	Wenn der Nutzer eine Funktion ausgelöst hat, soll die Applikation ihn nach der Bearbeitung darüber informieren, ob die Funktion erfolgreich ausgeführt wurde oder nicht.
U-34	DN-EK-01	Beachten von Gestaltungs-Best Practices	Wenn für die Gestaltung einer Funktion Best Practices existieren, soll die Applikation diese Best Practices einhalten.
U-35	DN-LE-01	Gut lesbarer Text	Die Applikation soll für Fließtext einen Schriftgrad zwischen 9 und 12 Pixel verwenden. Die Applikation soll Schrift mit einem hohen, aber nicht absoluten Farbkontrast gegenüber dem Hintergrund darstellen. Die Applikation soll Schriften verwenden, die auf allen Betriebssystemen verfügbar sind.
U-36	DN-LE-02	Verständlicher Text	Zeigt die Applikation Informationen in Form von Text an, soll die Applikation Begriffe verwenden, die die Nutzer verstehen.
U-37	DN-HI-01	Vorliegen einer Hilfestellung	Die Applikation soll eine Hilfestellung anbieten.

Tabelle 6: Usability-Anforderungsspezifikation für Dinner Now
Quelle: Eigene Darstellung

6.3 Weiterentwicklung des Fallbeispiels

Mit den erstellten Usability-Anforderungen und der beschriebenen Grundfunktionalität von Dinner Now wird im Folgenden eine mögliche Gestaltungsvariante erarbeitet. Ziel ist es, den vorhandenen Prototypen weiterzuentwickeln und mit Hilfe der Anforderungen dessen Benutzbarkeit zu verbessern. Dafür wird zunächst der neue

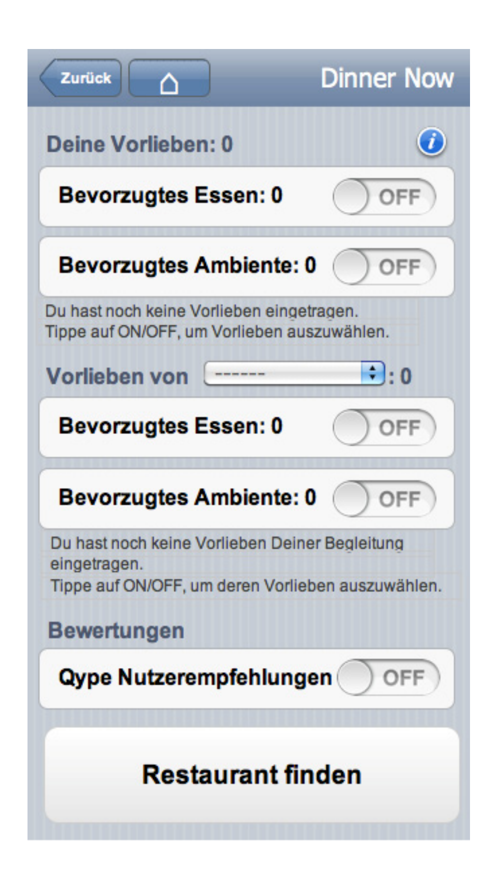
Grundaufbau der Bedienoberfläche beschrieben. Darauf folgt eine geordnete Bearbeitung der Usability-Anforderungen. Bei den Gestaltungsvorschlägen ist zu beachten, dass die Ästhetik der App noch nicht berücksichtigt wurde. Dementsprechend fehlen typische Elemente eines ansprechenden Interfacedesigns wie eigene Farben, Schriften, Illustrationen oder ein Logo.

Die Original-Entwürfe von Dinner Now zeigen das Auswählen der zu berücksichtigenden Vorlieben, einen Restaurant-Vorschlag und eine Geh-Route. Nicht vorhanden waren ein Startbildschirm und eine Funktion, mit der die Vorlieben tatsächlich eingegeben werden können. Diese beiden Bereiche wurden für die Weiterentwicklung neu erstellt. Im Folgenden sind der Startbildschirm, die Auswahl der Suchfaktoren, das Eingeben von Lieblingsessen und -ambiente, der neue Restaurant-Vorschlag und die Geh-Route zu sehen. Auf dem Startbildschirm entscheidet sich der Nutzer für eine Restaurant-Suche unter Berücksichtigung seiner Vorlieben oder für das Anzeigen des nächstgelegenen Restaurants. Der „Einstellungen“-Button auf dem Startbildschirm führt zu sonstigen technischen Einstellungsmöglichkeiten wie den aktivierten Datendiensten des Geräts und wird hier nicht weiter betrachtet.

Wählt er die Suche nach Vorlieben, kommt er zu dem Einstellungsmenü, das auch im Ursprungsentwurf zu sehen war. Hier wählt er aus, ob sein bevorzugtes Essen, sein bevorzugtes Ambiente und die gleichen Vorlieben seiner Begleitung bei der Suche berücksichtigt werden sollen. Er kann jetzt außerdem mit einem Drop-Down-Menü auswählen, wer ihn gerade begleitet. Sobald er einen der ON/OFF-Buttons antippt, wechselt der Dialog zu einer genauen Auswahl des bevorzugten Essens oder des bevorzugten Ambientes.

Mit „Restaurant finden“ werden wie zuvor Restaurant-Vorschläge aufgerufen und von dort Geh-Routen angewählt. Die neuen Vorschläge und Routen – zu sehen auf Abbildung 5 – ähneln dem ursprünglichen Entwurf, ordnen die Elemente jedoch anders an. Bei den Restaurant-Vorschlägen wird nach wie vor angezeigt, inwieweit diese den ausgewählten Vorlieben entsprechen. Der durchschnittliche Fit wurde ersetzt durch eine Tabelle, die jede Übereinstimmung mit einem Haken markiert. Es wurde außerdem ein weiterer Button eingefügt, der ein Wechseln zum vorherigen Vorschlag ermöglicht.





Zurück
Dinner Now
Wähle Dein bevorzugtes Essen:
Bevorzugtes Essen: 0
ON
Afrikanisch
Asiatisch
Deutsch
Französisch
Indisch
Italienisch
Mexikanisch
Mit sozialen Netzwerken verbinden
Weiter zu Vorlieben

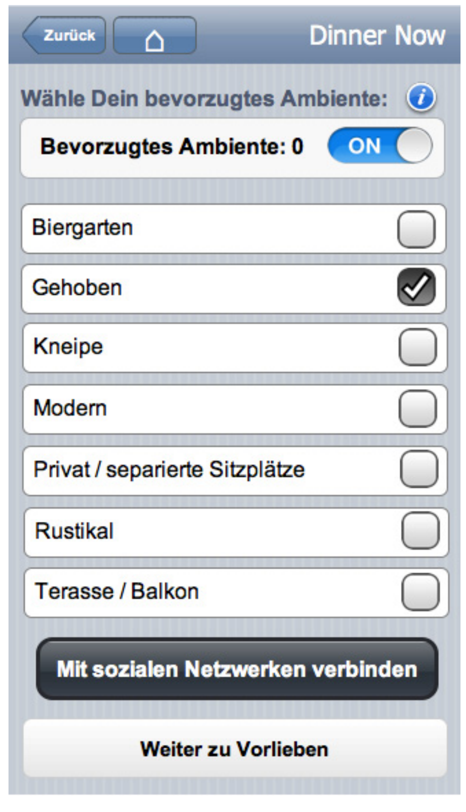
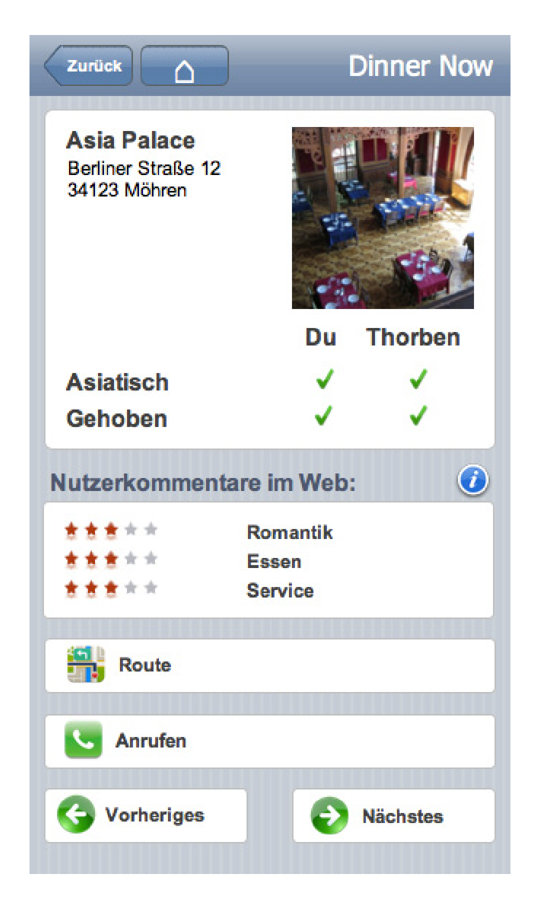


Abbildung 4: Weiterentwickelter Prototyp Dinner Now
Quelle: Eigene Darstellung



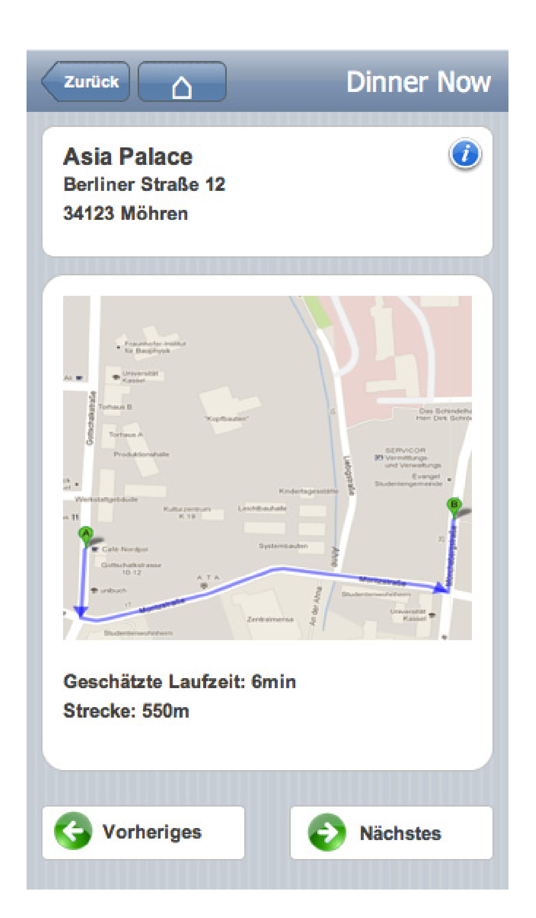


Abbildung 5: Veränderter Vorschlag und Route
Quelle: eigene Darstellung

Die erste Usability-Anforderung besagte, dass die Applikation nur Funktionen ausführen solle, die der Nutzer mit seinen Aktionen ausgelöst hat. Wenngleich automatische Prozesse in einem statischen Prototypen schwer darzustellen sind, sieht der jetzige Entwurf keine automatischen, selbstständigen Funktionen vor. DN-KO-02 verlangte, dass die Applikation dem Nutzer ermöglichen solle, sich innerhalb der Dialoge vor- und zurückzubewegen. Diese Anforderung wird mit mehreren Steuerelementen bedient: Auf dem Startbildschirm, der den Eintrittspunkt in die App darstellt, führen alle Elemente in der Navigation vorwärts. In jedem danach folgenden Dialog befindet sich links oben ein Zurück-Button, der den Nutzer zum zurückliegenden Dialog leitet sowie ein Home-Button (mit einem Haus-Symbol), der den Nutzer zum Startbildschirm bringt. In den Einstellungsmenüs führen die unteren Buttons „Weiter zu Vorlieben“ und „Restaurant finden“ stets vorwärts zur nächsten Funktion. Bei den Restaurant-Vorschlägen und den Geh-Routen geben die Buttons „Nächstes“ und „Vorheriges“ die Möglichkeit, zwischen vorherigen und kommenden Vorschlägen zu wechseln. Durch diese Buttons wird dem Nutzer eine vollständige Navigation geboten. Im Ursprungsentwurf fehlten ein Button, der zum Startbildschirm

zurück führt sowie ein Button zum vorherigen Vorschlag bei den Restaurant-Vorschlägen.

DN-KO-03 forderte Abbruchsmöglichkeiten, wenn die App eine Funktion ausführt. Die einzigen Funktionen, die eine längere Bearbeitungszeit haben könnten, sind das Laden der Vorschläge und das Laden der Routen. Im Folgenden sind die jeweiligen Lade-Bildschirme zu sehen:

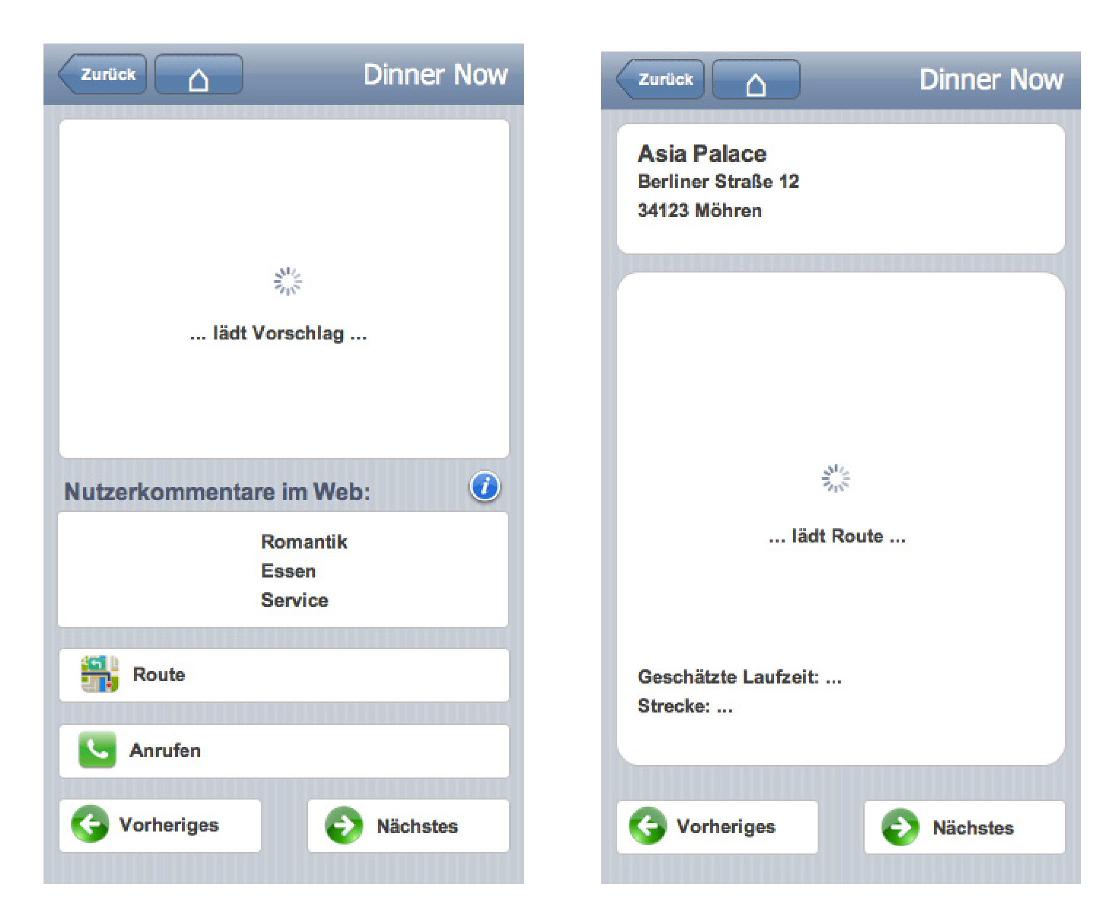


Abbildung 6: Ladebildschirme für das Laden von Restaurant-Vorschlägen und Routen
Quelle: Eigene Darstellung

Bei beiden Ladebildschirmen kann der Nutzer mit Hilfe der zuvor genannten Navigationsbuttons die Bearbeitung abbrechen und zum vorherigen Dialog zurückkehren. Die Punkte-Räder in der Mitte sollen animiert sein, damit deutlich wird, dass die Applikation aktiv arbeitet.

DN-KO-04 und DN-KO-05 beschäftigen sich mit dem Anlegen von Sucheinstellungen und dem Hinweis auf eben diese. Abbildung 7 zeigt die Auswahl der Sucheinstellungen. Im ersten Bild sind keine Sucheinstellungen aktiviert; alle möglichen Vorlieben sind mit „OFF" markiert. Ein Name für die Begleitung wird nicht angezeigt, da in dem Drop-Down-Feld noch keine Begleitung ausgewählt wurde

(„Vorlieben von ---- : 0“). Dadurch, dass alle Vorlieben offensichtlich ausgeschaltet sind, weiß der Nutzer, dass seine Restaurant-Suche keinerlei Vorlieben berücksichtigen würde.

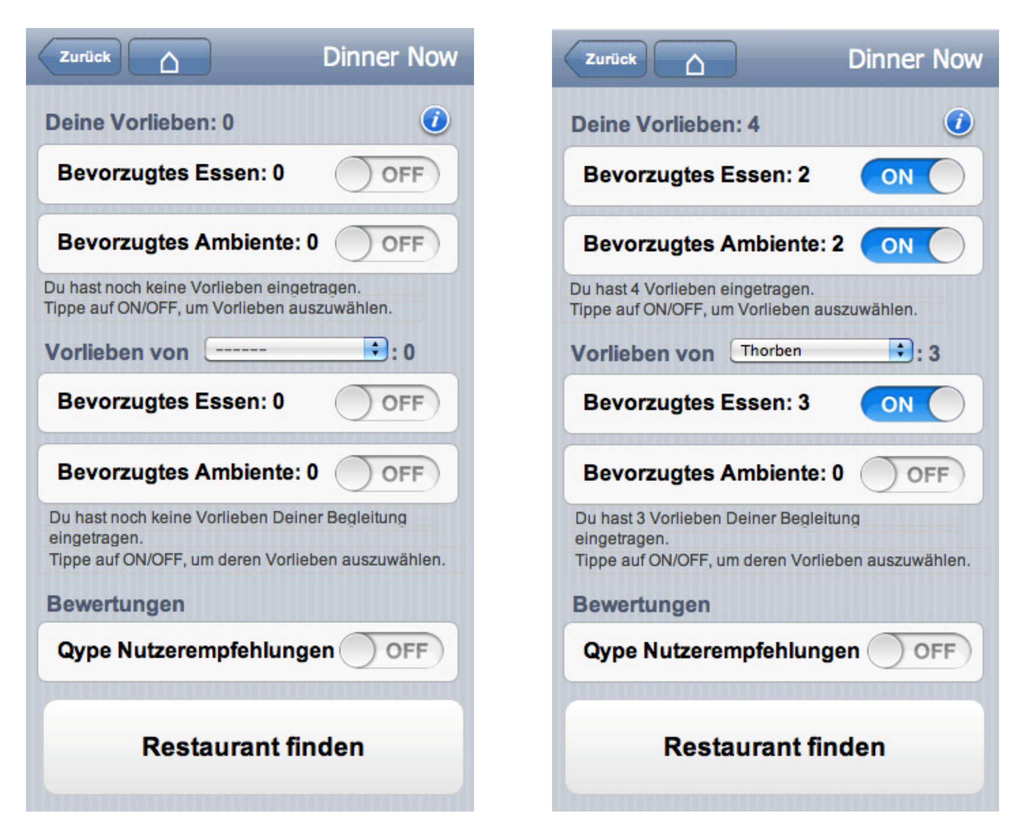


Abbildung 7: Deaktivierte und aktivierte Vorlieben
Quelle: Eigene Darstellung

Sobald der Nutzer einen der OFF-Regler auf ON stellt, wird ein Menü für die entsprechende Vorliebe ausgewählt. Dort kann er sein Lieblingsessen, sein bevorzugtes Ambiente oder entsprechende Vorlieben für seine Begleitung auswählen. Im vorliegenden Vorschlag bleiben einmal eingegebene Vorlieben des Nutzer erhalten, sobald er die App verlässt, ebenso wie die Vorlieben der Begleitung. Zusätzlich kann der Nutzer Vorlieben für verschiedene Begleitungen mit deren Namen abspeichern. Diese Funktion könnte mit der Anbindung von sozialen Netzwerken wahrscheinlich bequemer gestaltet werden; dies kann hier jedoch nur angedeutet werden. Durch die Einstellungsmenüs werden die Einstellungs-Anforderungen DN-KO-04 und DN-KO-05 bedient.

DN-KO-06 verlangte, dass die Restaurant-Suche einmal mit Vor-Einstellungen und einmal mit allen Einstellungen vorhanden sein soll. Auf dem Startbildschirm gibt es die Funktionen „Suche mit Vorlieben“ und „Das nächste Restaurant“. Die erste

Funktion führt zu den zuvor beschriebenen Einstellungen und stellt den Weg der vollen Kontrolle dar. Die zweite Funktion zeigt das nächstgelegene Restaurant mit räumlicher Nähe als einzigem Kriterium. Sie stellt den direkten, einfachen und bequemen Weg der Restaurant-Suche dar, da die App hier die Kontrollmöglichkeiten des Nutzers überspringt.



Abbildung 8: Vorlieben des Nutzers und seiner Begleitung
Quelle: Eigene Darstellung

DN-EF-01 forderte, dass die Bedienelemente der App weder von den Restaurant-Vorschlägen noch von den Routen ablenken. Durch die separaten, am Rand gelegenen Schaltflächen und der Größe der Inhalte wird diese Anforderung sowohl in den alten als auch den neuen Entwürfen erfüllt. Die Anforderung DN-EF-02 besagt, dass der Nutzer nur Aktionen ausführen soll, die seine Entscheidung erfordern. Diese Anforderung wird dadurch erfüllt, dass jede mögliche Aktion entweder das Verwenden einer Funktion oder einen direkten Wechsel zwischen den Dialogen darstellt – beide Arten von Aktionen erfordern stets eine Entscheidung des Nutzers. Weiterhin sollte durch DN-EF-03 der körperliche Einsatz des Nutzers minimiert werden. Fast alle Aktionen von Dinner Now lassen sich durch Tippen oder ein kurzes Streichen ausführen. Längeres Scrollen, Streichen oder Ziehen tauchen in der App

nicht auf. Sofern man diese Bewegungen als aufwendig betrachtet und Tippen als bequem, wäre die Anforderung als erfüllt zu betrachten.

DN-EF-04 forderte verständliche Anweisungen, wenn der Nutzer zu Aktionen aufgefordert wird. Die einzigen Anweisungen sind die Textanweisungen bei der Auswahl der Vorlieben und bei den Fehlerbenachrichtigungen. Sie sind jeweils mit kurzen Hauptsätzen formuliert, die als verständlich zu betrachten sind. Gemäß DN-EF-05 soll sich die Restaurant-Suche in überschaubare Teilschritte aufteilen. Es wird im vorliegenden Entwurf unterschieden in das Eingeben von Vorlieben, das Aktivieren von Vorlieben und das Durchführen der Suche, womit dieser Anforderung Rechnung getragen wird. In DN-EF-06 sollen zu Gunsten der Übersichtlichkeit immer nur aufgabenbezogene Informationen angezeigt werden. Da auf jedem Bildschirm entweder Inhalte oder Bedienelemente angezeigt werden, kann diese Anforderung sowohl bei den ursprünglichen als bei den neuen Entwürfen als erfüllt betrachtet werden. Es wird in Anforderung DN-EF-07 erbeten, dass Nutzer sich zwischen den einzelnen Schritten keine Informationen merken müssen. Als Informationen werden hier vorrangig die Einstellungen der Suche betrachtet. Die Nutzer können mit maximal zwei Finger-Berührungen stets zwischen ihren Vorlieben als Information und den Vorschlägen als Aufgabenergebnis wechseln und sehen in den Vorschlägen die Übereinstimmung mit ihren Vorlieben. Bei der Auswahl der zu berücksichtigenden Vorlieben sehen Nutzer, wie viele Vorlieben in jeder Kategorie angewählt wurden. In den Restaurant-Vorschlägen sehen sie, welchen ihrer Vorlieben das Restaurant entspricht. Ein Merken-Müssen von Informationen zwischen den Schritten ist daher nicht nötig.

Als einzige Anforderung zum Bereich Individualisierung fordert DN-IN-01, dass Nutzer aktiv Einstellungen an der Restaurant-Suche durchführen können. Diese Anforderung wurde bereits in den ursprünglichen Entwürfen berücksichtigt und zuvor ausreichend beschrieben.

Im Bereich der Fehlerhandhabung sollen durch Anforderung DN-FE-02 unmittelbare, hilfreiche Fehlerbenachrichtigungen eingebaut werden. Da die App in den Einstellungen der Suche nur mit Ein-Aus-Schaltflächen arbeitet, sind dort keine Fehler möglich. In der Restaurantsuche kann es passieren, dass der Nutzer diese auslöst, ohne Vorlieben eingegeben zu haben. Des Weiteren kann es vorkommen, dass keine Funk- oder Ortungsdienste zur Verfügung stehen, um die Funktion zu nutzen. In den

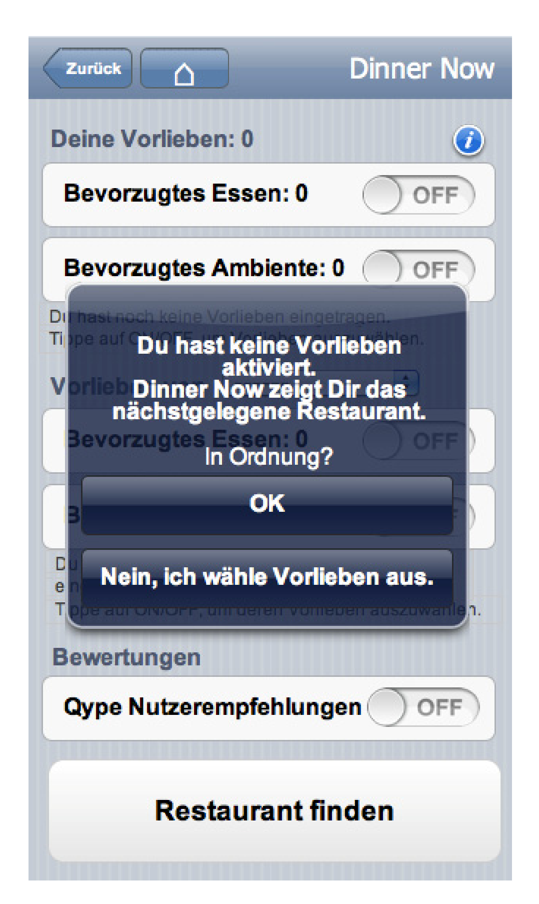
folgenden Abbildungen sind entsprechende Fehlerbenachrichtigungen zu sehen, die als Overlay-Fenster über der Navigation erscheinen. Die Hinweise sind mit einfachen Hauptsätzen formuliert. Als Optionen stehen jeweils ein Abbrechen und ein Beheben des Fehlers zur Verfügung, womit die Anforderung bedient sein sollte.

Durch DN-EF-03 soll sichergestellt sein, dass stets alle Eingaben in der App rückgängig gemacht werden können. Da der vorliegende Entwurf keine unwiderruflichen Eingaben erlaubt, ist diese Anforderung erfüllt. Weiter sollen durch DN-FE-04 Falscheingaben isoliert korrigiert werden können. Als Falscheingaben wären in diesem Sinne irrtümlich ausgewählte Vorlieben anzusehen. Diese können bereits jederzeit einzeln ein- und ausgeschaltet werden, weshalb auch diese Anforderung erfüllt ist.

DN-EP-01 und -02 erbitten einen offensichtlichen Startpunkt für die App, der jederzeit erreicht werden kann. Der Startbildschirm wurde bereits gezeigt, ebenso wie der Home-Button, der jederzeit zum Startbildschirm zurückführt. Barrieren finden sich im Startpunkt keine; der erste Restaurant-Vorschlag kann mit nur einem Fingertippen aufgerufen werden, auch die Vorlieben-Einstellungen sind nur ein Tippen entfernt. DN-EP-03 ist damit erfüllt. DN-EP-04 fordert zusätzlich, dass der Startpunkt einen Überblick über die Funktionen der App enthält. Es sind zwei Buttons zu sehen, die zu der Suche mit Einstellungen und der bequemen Einfach-Suche führen. Der Funktionsumfang wird damit dargestellt. Ein Eintrittspunkt muss außerdem einer Reihe ästhetischer Anforderungen genügen (siehe Kapitel 4.5), die jedoch nicht technisch formuliert werden konnten. Der hier gezeigte Entwurf sollte in der nächsten Überarbeitung eine ansprechende Gestaltung mit einladenden Bildelementen und einem stimmigen Logo erhalten.

Im Bereich der Wegfindung wird zunächst gefordert, dass die Applikation zu jedem Zeitpunkt anzeigt, wo der Nutzer sich befindet. Zunächst wird in der oberen Leiste stets der Name der App eingeblendet. Da der Entwurf das Standard-Menüdesign eines frühen iOS-Betriebssystems nutzt, weist dieser Titel auf das Nutzen einer App hin. In den Einstellungsmenüs wird mit Hilfe von Überschriften angezeigt, welche Vorlieben aktuell bearbeitet werden und zu wem diese gehören. Bei den Vorschlägen und den Routen soll anhand des gezeigten Inhalts klar werden, wo sich die Applikation aktuell befindet. Um hier mehr Platz für den Inhalt zu lassen, wird auf Überschriften verzichtet. Da die App ein Restaurant-Finder ist und die Vorschläge deren

Hauptfunktion sind, kann davon ausgegangen werden, dass an dieser Stelle kein Verirren seitens der Nutzer entsteht. Gemäß DN-WE-02 sollen die Bedienelemente stets aussagekräftige Titel haben. Die Bedienelemente zum Navigieren verwenden kurze Satzteile und sind genauso wie die dazugehörigen Funktionen formuliert: "Suche mit Vorlieben“, „Das nächste Restaurant“, „Zurück“, „Weiter zu Vorlieben“ und „Restaurant finden“ beschreiben das, was tatsächlich passiert. Die Bedienelemente sind daher aussagekräftig.



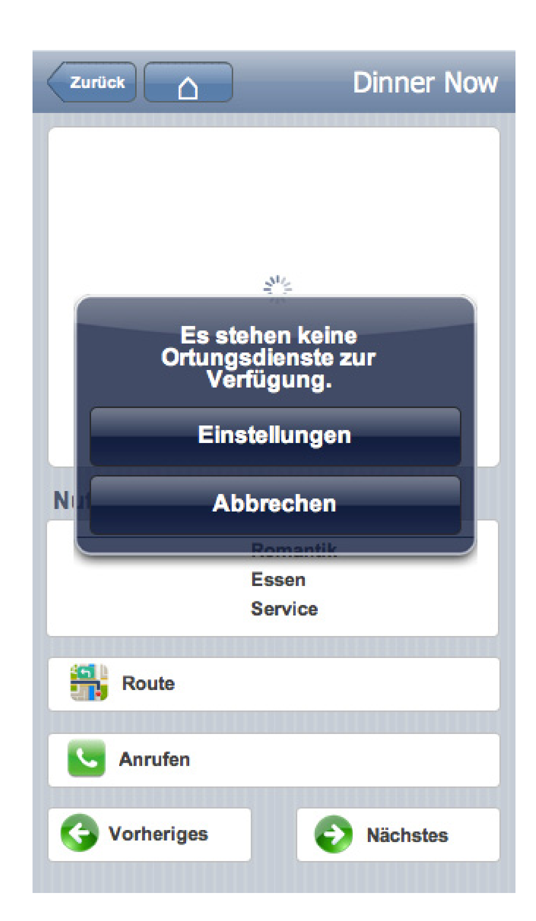


Abbildung 9: Fehlerbenachrichtigungen
Quelle: Eigene Darstellung

Der nächste Bereich widmet sich der Anordnung von Informationen. Zuerst sollen mit DN-AI-01 wichtige Informationen gegenüber weniger wichtigen hervorgehoben werden. Als wichtig werden hier vor allem jene Elemente betrachtet, mit denen die Funktionen ausgeführt werden. Dies sind die auszuwählenden Vorlieben und die Steuerelemente der Applikation. Ebenfalls wichtig sind die erzeugten Inhalte. In den Einstellungsmenüs haben die Buttons und Auswahlfelder, die den Kern der Funktionalität darstellen, weiße Balken und dunkelgraue, kontrastreiche Schrift. Sie treten dadurch deutlicher hervor als die Überschriften und sind somit hervorgehoben. Bei den Restaurant-Vorschlägen und den Routen sind v. a. die gezeigten Restaurants,

die Übereinstimmung mit den Vorlieben und die Karten wichtige Informationen. Sie sind deutlich größer dargestellt als die anderen Bildelemente. Gegenüber den ursprünglichen Entwürfen wurden die Schriften und die Restaurantvorstellung vergrößert. Mit DN-AI-02 sollen Informationen in der Darstellung sinnvoll gruppiert werden. Die Gruppierung in den Einstellungen ist sortiert nach Person (Nutzer oder Begleitung) sowie Vorliebe (Essen oder Ambiente). Eine andere Sortierung erscheint nicht sinnvoll, weshalb diese vom Ursprungsentwurf übernommen wird. Mit DN-AI-03 soll weiterhin sichergestellt sein, dass bei allen dargestellten Informationen die Navigation stets erreichbar bleibt. Durch die permanente obere Leiste mit dem Zurück- und dem Home-Button und den unteren „Vorwärts"-Buttons ist diese Anforderung erfüllt.

Die erste Anforderung zum Bereich Konsistenz sollte sicherstellen, dass alle Teile der App identisch formatiert sind und damit ein einheitliches Bild erzeugen. Dies wird einerseits dadurch erreicht, dass – wie in den ursprünglichen Entwürfen – immer gleiche oder ähnliche Schaltflächen und Schriften verwendet werden. Im Gegensatz zum ursprünglichen Prototypen wurden die Formulierungen angepasst: bei den Restaurant-Vorschlägen wurde die Übereinstimmung des Vorschlags mit den Vorlieben als „Fit" und „Nationalität" benannt, während im Vorlieben-Menü von bevorzugtem Essen die Rede war. Dies wird im neuen Vorschlag mit einer einfachen Tabelle gelöst, die die gleichen Wörter wie bei den Sucheinstellungen verwendet und eindeutig Übereinstimmungen und Abweichungen anzeigt. Die konsistente Darstellung aller Bereiche ist damit erfüllt. Es wird in DN-CO-02 gefordert, dass die App auf unterschiedlichen Endgeräten identisch formatiert ist. Diese Anforderung kann mit einem statischen Prototypen wie dem hier Verwendeten nicht behandelt werden.

Anforderung DN-SI-01 besagte, dass die App stets ihren Systemstatus anzeigen solle. Ziel ist hierbei, dass Nutzer stets wissen, was die App gerade tut und welche Optionen ihnen offenstehen. Befindet sich der Nutzer auf dem Startbildschirm, in den Sucheinstellungen oder bei den Vorschlägen, wartet die App lediglich auf Eingaben. Ein Anzeigen des Systemstatus ist hier nicht notwendig. Da statische Informationen angezeigt werden, wird der Nutzer nicht damit rechnen, dass die App arbeitet. Ungewissheit kann lediglich bei den Ladebildschirmen aufkommen. Hier wird neben dem animierten Punkte-Rad die Nachricht „... lädt Vorschlag ..." oder „... lädt Route ..." angezeigt. Dies verdeutlicht dem Nutzer, dass die App gerade versucht,

Informationen zu verarbeiten. Zusammen mit den Fehlerbenachrichtigungen sollte der Systemstatus zu jedem Zeitpunkt erkennbar sein.

DN-RM-01 bis -04 beschäftigen sich mit der unmittelbaren Rückmeldung bei Aktionen durch den Nutzer. Dieses sofortige Reagieren lässt sich in einem statischen Prototypen nicht darstellen. Die App soll so funktionieren, dass alle im Prototypen angebotenen Schaltflächen eine sofortige Reaktion in Form eines Häkchens, einer Auswahl oder einem Dialogwechsel auslösen. Für den Fall der längeren Bearbeitungszeit wird ein weiteres Mal auf die beiden Ladebildschirme verwiesen, die sofort erscheinen, sobald eine Suche ausgelöst oder eine Route aufgerufen werden. Sobald die entsprechenden Inhalte geladen wurden, erscheinen sie auf dem leeren Ladebildschirm. Die erfolgreiche Bearbeitung wird damit eindeutig signalisiert.

DN-EK-01 gibt vor, dass beim Design der App Best Practices eingehalten werden sollen. Konventionen sind unter anderem, dass sich der Zurück-Button in der oberen linken Ecke befindet oder dass die Buttons zum Auslösen einer Funktion unter den Einstellungsmöglichkeiten liegen. Diese Konventionen wurden im ursprünglichen und im neuen Prototypen eingehalten. Des Weiteren funktionieren iOS-Einstellungsmenüs oft so, dass sich bei Einschalten eines ON/OFF-Buttons ein Untermenü öffnet (vgl. Apple 2013, 54). Das gleiche Prinzip wurde bei den Vorlieben-Einstellungen verwendet. Die Benutzung der Suchfunktion sollte Nutzern also von dem Verwenden von iOS bekannt sein.

Für die Lesbarkeit des Textes wurden detaillierte Anforderungen übernommen und berücksichtigt: der Schriftgrad ist stets mindestens 10, die Schrift ist immer Arial (das Logo ausgenommen), welche überall verfügbar ist. Die Kontraste sind immer kräftig, wenngleich sie bei den weißen Schaltflächen besonders hoch sind. Der Kontrast ist hier allerdings nicht absolut, da statt schwarzer Schrift eine graue verwendet wird. Die Schrift sollte daher immer deutlich und angenehm zu lesen sein. Bei der in DN-LE-02 behandelten Verständlichkeit des Textes wurde darauf geachtet, stets einfache Hauptsätze mit eindeutigen, konsistenten Begriffen zu verwenden.

Der letzte Bereich ist das Vorliegen einer Hilfestellung durch DN-HI-01. Dies wird durch zwei Elemente gelöst. Die erste Hilfestellung sind die Text-Hinweise in der Auswahl der zu berücksichtigenden Vorlieben. Sie sagen dem Nutzer, ob Vorlieben berücksichtigt werden und was er tun muss, um die Einstellungen zu verändern. Derzeit verbrauchen diese Texte viel Platz. Zugunsten einer helleren, aufgeräumten

Oberfläche können diese Hinweise auch hinter einem Info-Button versteckt werden. Die zweite Hilfestellung sind kleine Buttons mit einem „i“, die bei jedem Dialog angezeigt werden sowie der „FAQ“-Button auf dem Startbildschirm. Hier können die zuvor genannten Hinweise oder eine beliebig ausführliche Bedienungsanleitung hinterlegt werden.

6.4 Ergebnisse aus der Anwendung der Usability-Anforderungsmuster

Der ursprüngliche Prototyp von Dinner Now berücksichtigte bereits viele gängige Usability-Empfehlungen und besaß eine schlichte, klare Bedienoberfläche. Das Prüfen sämtlicher Usability-Anforderungsmuster deckte jedoch Verbesserungspotenziale und Lücken auf. So stellt beispielsweise der Eintrittspunkt einen entscheidenden Benutzbarkeitsfaktor dar, der zunächst fehlte. Ihm kommt die Aufgabe zu, dem Nutzer die Hauptfunktionen der Applikation zu vermitteln und ihn zu ihrer Verwendung zu motivieren. Der neu gestaltete Startbildschirm enthält zwei große Buttons für die beiden Varianten der Restaurant-Suche sowie viel Raum für ansprechende Gestaltungselemente. Im Vergleich zu dem ursprünglichen Entwurf erhält der Nutzer bei dem neuen Prototypen wesentlich mehr Kontrolle sowie Hilfestellung beim Erfüllen der Aufgabe. Durch die Anwendung der Anforderungsmuster wurden zusätzliche Steuerelemente wie der „Vorheriges“-Button bei den Restaurant-Vorschlägen oder der ständig angezeigte Home-Button hinzugefügt. Außerdem erhält der Nutzer nun mehr Informationen durch die App. Er sieht, wie viele Vorlieben ausgewählt sind und welche Vorlieben bei einem Restaurant bedient werden. Er erfährt außerdem, was er tun muss, um die App zu verwenden, ohne dass er es durch Ausprobieren und langes Vor- und Zurückgehen herausfinden muss. Durch das Anbieten zweier Wege beim Finden eines Restaurants überlässt die App Konkurrenzprodukten keinen Vorteil: mit dem Tippen auf „Das nächste Restaurant“ werden wie bei anderen Restaurant-Findern ohne großen Bedienaufwand Vorschläge angezeigt.

Die Bearbeitung der Usability-Anforderungen an dem ursprünglichen Prototyp von Dinner Now deckte auf, an welchen Stellen dieser eine bessere Benutzbarkeit erhalten könnte. Das Prüfen sämtlicher Muster half dabei, die Navigationswege der App und die angezeigten Informationen deutlich zu erweitern. Die nicht verwendeten Muster wurden meist nur deshalb abgelehnt, weil die Funktionalität von Dinner Now relativ einfach und kurz ist. Der Katalog mit den Anforderungsmustern ist geeignet, auf ein

reales Software-Projekt angewendet zu werden. Mit den Mustern können passende, angemessene Usability-Anforderungen an eine Applikation gestellt werden. Der Nutzen des Katalogs liegt dabei vor allem in dessen inhaltlicher Breite. Es ist zu erwarten, dass geübte Entwickler und Designer stets eine Reihe von Faktoren beachten, die zu guter Benutzbarkeit führen. So besaß auch der ursprüngliche Prototyp von Dinner Now eine verständliche, klare Bedienoberfläche. Das Anwenden eines Leitfadens wie dem Katalog mit Usability-Anforderungsmustern hilft jedoch, das vollständige Usability-Potenzial einer Software aufzudecken. Wenn auch nicht alle Muster des Katalogs zum Einsatz kommen, schafft das Prüfen aller Muster ein Bewusstsein für die verschiedenen Möglichkeiten, Usability zu erzeugen. Entwickler laufen bei Verwendung des Katalogs nicht Gefahr, kleine, aber hilfreiche Gestaltungselemente auszulassen oder zu vergessen. Stattdessen führt der Katalog zum Überdenken von Gestaltungsfeldern, die teilweise sogar mit der Funktionalität zusammenhängen. Beispielsweise setzte sich der ursprüngliche Prototyp nicht mit möglichen Fehlern auseinander. Das Einbauen von aussagekräftigen Fehlermeldungen schaffte jedoch ein Bewusstsein dafür, was bei der Applikation aus welchem Grund nicht funktionieren könnte und wie dies dem Nutzer erklärt werden könne. Die Usability-Anforderungsmuster stellen somit eine sinnvolle Ergänzung für das Erstellen von funktionalen Anforderungskatalogen dar.

In einem nächsten Test kann geprüft werden, ob mit Hilfe der Anforderungsmuster die wahrgenommene Usability verbessert werden kann. Dazu könnte ein Prototyp einem Usability-Test unterzogen werden, ehe er mit Hilfe der Muster verändert wird. Nach der Veränderung müsste ein zweites Mal mit Hilfe eines Usability-Tests herausgefunden werden, ob sich die Schnelligkeit der Aufgabenerfüllung und die wahrgenommene Zufriedenheit mit dem Prototypen verbessert haben.

7 Fazit und Ausblick

Ziel der vorliegenden Arbeit war es, einen Katalog mit Usability-Anforderungsmustern zu erstellen, der einerseits theoretisch fundiert und andererseits praktisch anwendbar ist. Die erste Forschungsfrage bestand in dem Erarbeiten der Anforderungsmuster auf Basis einer Literaturrecherche. Es zeigte sich, dass eine große Vielfalt an Gestaltungsmöglichkeiten besteht, um gute Benutzbarkeit zu schaffen. Viele dieser Gestaltungswege haben den Charakter von Lösungsvarianten und eignen sich nicht dazu, Anforderungen an ein System zu stellen. Die beiden Bereiche, für die keine Anforderungsmuster mit Prüfmechanismus gefunden werden konnten, sind Ästhetik und Metaphern. In den anderen geprüften Themenbereichen wurde mindestens ein Anforderungsmuster formuliert. Diese 13 Bereiche behandeln sehr unterschiedliche Gestaltungsfelder. Sie reichen von der bloßen Darstellung von Elementen über die Abläufe der Funktionen bis zu den Informationen, die das System dem Nutzer gibt. Das Gegenüberstellen der einzelnen Muster lies erkennen, dass viele Muster auf die gleichen Usability-Ziele hinarbeiten. Dies deckt sich mit den eingangs genannten Definitionen von Usability. Darin wurde Benutzbarkeit als Potenzial beschrieben und nicht als Systemeigenschaft. Usability wird somit nicht dadurch erreicht, dass eine bestimmte Zahl der erstellten Anforderungsmuster berücksichtigt wird. Das Potenzial kann aber deutlich gesteigert werden, je mehr der genannten Bereiche geprüft und angewendet werden. Die erstellten 47 Anforderungsmuster können dabei gut in Kombination eingesetzt werden. Es wurden nur drei potenzielle Konflikte zwischen einzelnen Mustern gefunden. Die Usability-Anforderungsmuster eignen sich daher gut dazu, bestehende Anforderungskataloge zu ergänzen, ohne dem Projekt ein neues Konfliktfeld hinzuzufügen.

Für das Beantworten der zweiten Forschungsfrage, die auf die Anwendbarkeit und Qualität der Muster zielte, wurden Gespräche mit Praktikern geführt. Die Befragten begrüßten das Anlegen eines umfassenden Katalogs mit Usability-Empfehlungen und bestätigten, dass der Katalog alle bekannten Usability-Bereiche abdeckt. Er wurde als Möglichkeit betrachtet, bestehende Anforderungsspezifikationen und Designs auf Usability-Aspekte hin zu überprüfen. Da alle befragten Betriebe Usability bereits in anderer Form bearbeiten, wurde die direkte praktische Anwendbarkeit der Anforderungsmuster jedoch nicht bestätigt. Dass außerdem lediglich drei Unternehmen befragt wurden und diese keine Anforderungsmuster verwenden,

schränkt die Aussagekraft der Gesprächsergebnisse ein. Hier sind weitere Befragungen bei Unternehmen notwendig, die nicht nur Anforderungen, sondern auch Anforderungsmuster einsetzen. Die Gespräche ergaben, dass bei den Befragten kein alternatives Instrument vorliegt, um Usability im Entwicklungsprozess greifbar zu dokumentieren und zu verfolgen. Zwar stellt Benutzbarkeit stets ein zentrales Entwicklungsziel dar, jedoch verlassen sich die Befragten auf ihre Erfahrungen und ihre Gestaltungskompetenz in diesem Bereich. Da sie auch keine Usability-Tests durchführen, findet offenbar keine Überprüfung der Benutzbarkeit vor Markteintritt statt. Die erstellten Anforderungsmuster können hier die Aufgabe übernehmen, ein standardisiertes Usability-Wissen aufzubauen und den erreichten Grad an Benutzbarkeit messbar zu machen. Das Inhaltsverzeichnis des Muster-Katalogs kann selbst für erfahrene Entwickler als Checkliste dienen, um ihre Arbeit auf Vollständigkeit zu überprüfen. Unerfahreneren Entwicklern kann der Katalog als Leitfaden dienen, um wichtige Gestaltungshinweise zu erhalten. Ein weiteres Forschungsthema kann darin bestehen, die optimale Form der Usability-Anforderungsmuster zu finden. Die befragten Unternehmen setzen neben ihren Anforderungen unterschiedliche Arbeitshilfsmittel ein, beispielsweise Richtlinien für die Programmierung oder technische Checklisten. Um die Benutzbarkeit der Usablity-Anforderungsmuster zu erhöhen, könnten verschiedene Formen der Muster wie beispielsweise Online-Checklisten erprobt werden. Würden die erarbeiteten Empfehlungen nicht in Form von Anforderungsmustern, sondern in einfacherer Form angeboten, ließe sich damit ein größerer Benutzerkreis erschließen.

Mit der letzten Forschungsfrage sollte geprüft werden, ob sich die Muster an einem konkreten Fallbeispiel anwenden lassen, um sowohl Anforderungen als auch Gestaltungsvorschläge aufzustellen. Bei der Betrachtung der Restaurant-Finder-App „Dinner Now" kamen 37 der 47 Muster zum Einsatz. Mit den erstellten Anforderungen wurden v. a. die Steuermöglichkeiten, der Eintrittspunkt und die angezeigten Informationen des bearbeiteten Prototyps erweitert. Die Muster konnten somit erfolgreich angewendet werden. Dabei wurde deutlich, dass nicht alle Muster angewendet werden müssen, um eine ausreichende Usability zu erreichen. Das Bearbeiten sämtlicher Muster führte aber zum Berücksichtigen von Themenbereichen, die der ursprüngliche Prototyp nicht enthielt. Hier sind vor allem die fehlende Startseite und die fehlenden Fehlermeldungen zu nennen. Das Überprüfen des vollständigen Muster-Katalogs schuf somit ein Bewusstsein für alle denkbaren Usability-Problembereiche. Der Checklistencharakter, den die befragten Entwickler in

dem Katalog erkannten, bestätigte sich bei der Anwendung der Muster an dem Fallbeispiel.

Die Arbeit belegt, dass Usability in Form von Anforderungen in die Entwicklung von Software-Applikationen aufgenommen werden kann. Da Usability ein vielseitiges Problemfeld ist, empfiehlt sich das Anwenden eines Usability-Leitfadens, um alle denkbaren Problembereiche zu berücksichtigen. Bisher existieren unzählige Gestaltungsmöglichkeiten, um gute Benutzbarkeit zu schaffen. Die Gestaltungsmöglichkeiten stellen jedoch Lösungen dar, die erst zur Anwendung kommen, wenn die zu lösenden Probleme bekannt sind. Der erstellte Katalog mit Anforderungsmustern kann auf diese Problembereiche aufmerksam machen, bevor das System einen Reifegrad erreicht, der Änderungen in der Benutzeroberfläche nur schwer zulässt. Er kann Entwicklern Denkanstöße für das geben, was ein System neben seinen eigentlichen Funktionen können soll. Gleichzeitig schränken die Usability-Anforderungen die Gestaltung nur in geringem Maße ein, sie geben stattdessen Hinweise zur Erweiterung von Designs. Ob Usability-Anforderungsmuster in realen Projekten eingesetzt werden können, hängt maßgeblich damit zusammen, ob die Entwickler bereits mit Anforderungsmustern arbeiten. Um die praktische Anwendbarkeit der Muster weiterzuentwickeln, sollten zusätzliche Expertengespräche mit Anwendern von Anforderungsmustern geführt werden. Die Existenzberechtigung der Usability-Anforderungsmuster wurde belegt, weitere Forschung sollte sich mit der Benutzbarkeit der Muster beschäftigen.

Literaturverzeichnis

Alby, T. (2008): Das mobile Web. Carl Hanser, München 2008.

Apple Inc. (2013): iPhone Benutzerhandbuch. In: http://www.apple.de, zugegriffen am 20.11.2013.

Bennet, K. B.; Flach, J. M. (2011): Display and Interface Design. CRC Press, Boca Raton 2011.

Benyon, D. (2010): Designing Interactive Systems. 2. Aufl., Person, Harlow 2010.

Bevan, N. (2001): International standards for HCI and usability. In: International Journal of Human-Computer Studies, Vol. 55 (2001), S. 533-552.

Blair-Early, A.; Zender, M. (2008): User Interface Design Principles for Interaction Design. In: Design Issues, Vol. 24 (2008) Nr. 3, S. 85-107.

Böhringer, J.; Bühler, P.; Schlaich, P. (2011): Kompendium der Mediengestaltung. 5., überarb. u. erw. Aufl., Springer, Berlin-Heidelberg 2011.

Borenstein, N. S.; Thyberg, C. A. (1991): Power, ease of use in a coorperative work in a practical multimedia message system. In: International Journal of Man-Mashine Studies, Vol. 34 (1991), S. 229-259.

Brereton, P.; Kitchenham, B. A.; Budgen, D.; Turner, M.; Khalil, M. (2007): Lessons from applying the systematic literature review process within the software engineering domain. In: The Journal of Systems and Software, Vol. 80 (2007), S. 571-583.

Cappel, J. J.; Huang, Z. (2007): A Usability Analysis of Company Websites. In: Journal of Computer Information Systems, Vol. 48 (2007) Nr. 1, S. 117-123.

Connel, B. R.; Jones, M.; Mueller, J.; Mullick, A.; Ostroff, E.; Sanford, J.; Steinfeld, E.; Story, M.; Vanderheiden, G. (1998): The Principles of Universal Design. In: The Universal Design File: Designing for People of All Ages and Abilities. Hrsg.: Joines, S.; Valenziano, S. NC State University, New York (1998), S. 34-84.

Ebert, C. (2012): Systematisches Requirements Engineering. 4., überarb. Aufl., dpunkt: Heidelberg 2012.

Franch, X.; Palomares, C.; Quer, C.; Renault, S.; De Lazzer, F. (2010): A Metamodel for Software Requirement Patterns. In: REFSQ 2010. Hrsg: Wieringa, R.; Persson, A. Springer, Heidelberg 2010, S. 85-90.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. (1995): Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Company, Reading 1995.

Harrison, R.; Flood, D.; Duce, D. (2013): Usability of mobile applications: literature review and rationale for a new usability model. In: Journal of Interaction Science, Vol. 1 (2013), Nr. 1, S. 1-16.

Harty, J. (2011): Finding usability Bugs within automated Tests. In: Queue, Vol. 9 (2011) Nr. 1, o.S.

Hassenzahl, M.; Monk, A. (2010): The Inference of Perceived Usability From Beauty. In: Human-Computer Interaction, Vol. 25 (2010), S. 235-260.

Hertzum, M. (2010): Images of Usability. In: International Journal of Human-Computer Interaction, Vol. 26 (2010) Nr. 6, S. 567-600.

Hix, D.; Hartson, H. R. (1993): Developing User Interfaces - Ensuring Usability Through Product & Process. John Wiley & Sons, New York 1993.

Hoffmann, A.; Hoffmann, H.; Leimeister, J. M. (2012): Anforderungen an Software Requirement Pattern in der Entwicklung sozio-technischer Systeme. In: 42. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Braunschweig, Germany.

Holzinger, A. (2005): Usability Engineering Methods For Software Developers. In: Communications oft he ACM, Vol. 48 (2005) Nr. 1, S. 71-74.

Hsu, Y. C.; Boling, E. (2007): An approach for designing composite metaphors for user interfaces. In: Behaviour & Information Technology, Vol. 26 (2007) Nr. 3, S. 209-220.

Hull, E.; Jackson, K.; Dick, J. (2011): Requirements Engineering. 3. Aufl., Springer, London 2011.

Huang, S.-C.; Chou, I.-F.; Bias, R. G. (2006): Empirical Evaluation of a Popular Cellular Phone's Menu System: Theory Meets Practice. In: Journal of Usability Studies, Vol. 1 (2006) Nr. 2, S. 91-108.

International Standards Organization (2006): EN ISO 9241: Ergonomie der Mensch-System-Interaktion. Internationaler Standard, Genf 2006.

Jordan, P. W. (1998): An Introduction to Usability. Taylor & Francis, London 1998.

Kaasinen, E. (2005): User acceptance of mobile services – value, eas of use, trust and ease of adoption. Diss., Tampere University of Technology.

Kim, K.; Jacko, J.; Salvendy, G. (2011): Menu Design for Computers and Cell Phones: Review and Reappraisal. In: International Journal of Human-Computer Interaction, Vol. 27 (2011) Nr. 4, S. 383-404.

Kitchenham, B.; Brereton, O. P.; Budgen, D.; Turner M. Bailey, J.; Linkman, S. (2009): Systematic literature reviews in software engineering – A systematic literature review. In: Information and Software Technology, Vol. 51 (2009), S. 7-15.

Knittle, D. L.; Ruth, S.; Gardner, E. P. (1986): Establishing User-Centered Criteria for Information Systems: A Software Ergonomics Perspective. In: Information & Management, Vol. 11 (1986) Nr. 4, S. 163-172.

Kurosu, M.; Kashimura, K. (1995): Apparent Usability vs. Inherent Usability. In: Chi '95 Conference Companion on Human Factors in Computing Systems, S. 292-293.

Lee, Y.; Chen, A. N. K. (2012): Usability Design ad Psychological Ownership of a Virtual World. In: Journal of Management Information Systems, Vol. 28 (2012) Nr. 3, S. 269-307.

Lidwell, W.; Holden, K.; Butler, J. (2003): Universal Principles of Design. Rockport Publishers, Beverly 2003.

Lin, H. X.; Choong, Y.-Y.; Salvendy, G. (1997): A proposed index os usability: A method for comparing the relative usability of different software systems. In: Baheviour & Information Technology, Vol. 16 (1997) Nr. 4-5, S. 267-277.

Ludewig, J.; Lichter, H. (2010): Software Engineering. 2., überarb. u. aktual. Auflage, dpunkt, Heidelberg 2010.

Mathieson, K.; Keil, M. (1998): Beyond the interface: Ease of use and task/technology fit. In: Information & Management, Vol. 34 (1998), S. 221-230.

Murray, K. B.; Häubl, G. (2010): Freedom of Choice, Ease of Use, and the Formation of Interface Preferences. In: MIS Quarterly, Vol. 35 (2011) Nr. 2, S. 955-976.

Nassar, V. (2012): Common criteria for usability review. In: Work: A Journal of Prevention, Assessment and Rehabilitation, Vol. 41 (2012) Nr. 1, S. 1053-1057.

Nielsen, J. (1993): Usability Engineering. Academic Press, San Diego 1993.

Nielsen, J. (2000): Designing Web Usability. New Riders Publishing, Indianapolis 2000.

Nielsen, J.; Mack, R. (1994): Usability Inspection Methods. John Wiley & Sons, New York.

Palanque, P.; Farance, C.; Bastide, R. (1999): Embedding Ergonomic Rules as Generic Requirements in a Formal Developement Process of Interactive Software. In: Human-Computer Interaction INTERACT ’99. Hrsg.: Sasse, M. A.; Johnson, C. IOS Press, Amsterdam 1999, S. 408-416.

Petter, A.; Khazanchi, D.; Murphi, J. D. (2010): A Design Science Based Evaluation Framework for Patterns. In: The DATA BASE for Advances in Information Systems, Vol. 41 (2010) Nr. 3, S. 9-26.

Pohl, K. (2008): Requirements Engineering. 2. Aufl., dpunkt, Heidelberg 2008.

Pohl, K.; Rupp, C. (2009): Basiswissen Requirements Engineering. dpunkt, Heidelberg 2009.

Preece, J. (2000): Online Communities. Designing Usability, Supporting Sociability. John Wiliey & Sons, West Sissex 2000.

Rampl. H. (2007): Handbuch Usability. In: http://www.handbuch-usability.de, zugegriffen am 27.08.2013.

Reeves, L. M.; Lai, J.; Larson, J. A.; Oviatt, S.; Balaji, T. S.; Buisine, S.; Collings, P.; Cohen, P.; Kraal; B.; Martin, J.-C.; Mc Tear, M.; Raman, T.; Stanney,

K. M.; Su, H.; Wang, Q. Y. (2004): Guidelines for multimodal user interface design. In: Communications oft he ACM, Vol. 47 (2004) Nr. 1, S. 57-59.

Renault, S.; Méndez, O.; Franch, X.; Quer, C. (2009): A pattern-based method for building requirements documents in call-for-tender processes. In: International Journal of Computer Science and Applications, Vol. 6 (2009) Nr. 5, S. 175-202.

Rosson, M. B.; Carrol, J. M. (2002): Usability Engineering. Morgan Kaufmann: San Francisco 2002.

Rupp, C. & SOPHIST GROUP (2009): Requirements-Engineering und –Management. 5., aktual. u. erw. Aufl. Carl Hanser, München 2009.

Scapin, D. L.; Bastien, J. M. C. (1997): Ergonomic criteria for evaluating the ergonomic quality of interactive systems. In: Behaviour & Information Technology, Vol. 16 (1997) Nr. 4-5, S. 220-231.

Söllner, M.; Hoffmann, A.; Hoffmann, H. & Leimeister, J. M. (2012): Vertrauensunterstützung für sozio-technische ubiquitäre Systeme. In: Zeitschrift für Betriebswirtschaft, Vol. 4 (2012), S. 109-140.

Stapelkamp, T. (2010): Interaction- und Interfacedesign. Springer, Heidelberg 2010.

Tractinsky, N. (1997): Aesthetic and Apparent Usability: Empirically Assessing Cultural and Methodical Issues. In: CHI ‘97 Proceedings oft he ACM SIGCHI Conference on Human factors in computing systems. ACM, New York 1997, S. 115-122.

Versteegen, G. (2004) (Hrsg.): Anforderungsmanagement. Springer, Heidelberg 2004.

Withall, S. (2007): Software Requirement Patterns. Microsoft Press, Washington 2007.

Wurhofer, D.; Obrist, M.; Beck, E.; Tscheligi, M. (2010): A Quality Criteria Framework for Pattern Validation. In: International Journal on Advances in Software, Vol. 3 (2010) Nr. 1-2, S. 252-264.

Xie, H. (2003): Supporting ease-of-use and user control: desired features and structure of Web-based online IR systems. In: Information Processing and Management, Vol. 39 (2003), S. 899-922.

Ziefle, M. (2002): The influence of user expertise and phone complexity on performance, ease of use and learnability of different mobile phones. In: Behaviour & Information Technology, Vo. 21 (2002) Nr. 5, S. 303-311.

Anhang

Anhang A Usability-Anforderungsmuster

Übersicht

A.1 Kontrolle und Einschränkungen 108

KO-01: Explizite Aktionskontrolle ... 108
KO-02: Steuerbarkeit durch Vorliegen einer Navigation ... 109
KO-03: Funktionen pausieren und abbrechen ... 110
KO-04: Informationen zu Vor-Einstellungen ... 111
KO-05: Vor-Einstellung abspeichern ... 112
KO-06: Zwei Wege zur Aufgabenbewältigung ... 113
KO-07: Kein Anbieten nicht-verfügbarer Funktionen ... 114

A.2 Effizienz 115

EF-01: Inhalt herausstellen ... 115
EF-02: Aufwand des Nutzers minimieren ... 116
EF-03: Körperlichen Einsatz minimieren ... 117
EF-04: Verständliche Anweisungen ... 118
EF-05: Aufteilen komplexer Aufgaben ... 119
EF-06: Übersichtlichkeit ... 120
EF-07: Kein Merken-Müssen von Informationen ... 121

A.3 Individualisierbarkeit 122

IN-01: Aktive Personalisierung anbieten ... 122
IN-02: Passive Personalisierung anbieten ... 123
IN-03: Mehrere Aktionen gleichzeitig ausführen ... 124

A.4 Fehler-Handhabung 125

FE-01: Bestätigung kritischer Aktionen ... 125
FE-02: Fehler-Benachrichtigung ... 126
FE-03: Aktionen rückgängig machen ... 127
FE-04: Isoliertes Korrigieren von falschen Eingaben ... 128
FE-05: Überprüfen von Eingaben vor Auslösen der Funktion ... 129
FE-06: Automatisches Speichern überschriebener Inhalte ... 130

A.5 Eintrittspunkte 131

EP-01: Startpunkt anbieten ... 131
EP-02: Jederzeitiges Zurückkehren zum Startpunkt ... 132

EP-03: Keine Barrieren im Startpunkt 133
EP-04: Überblick-Funktion des Startpunkts 134

A.6 Wegfindung 135

WE-01: Orientierung anbieten 135
WE-02: Aussagekräftige Bedienelemente 135

A.7 Anordnung von Informationen 136

AI-01: Wichtige Informationen hervorheben 136
AI-02: Informationen gruppieren 137
AI-03: Ständig erreichbare Navigation 138

A.8 Konsistenz 139

CO-01: Alle Teile einer Applikation ähnlich gestalten 139
CO-02: Konsistent gestaltete Funktionen 140
CO-03: Konsistent gestaltete Informationen 141
CO-04: Konsistente Darstellung auf verschiedenen Endgeräten 142

A.9 Sichtbarkeit des Systemstatus 143

SI-01: Systemstatus anzeigen 143

A.10 Rückmeldung 144

RM-01: Rückmeldung bei Aktionen 144
RM-02: Hinweis bei langer Bearbeitung 145
RM-03: Sofortiges Anzeigen von Eingaben 146
RM-04: Erfolgreiche Bearbeitung anzeigen 147

A.11 Erwartungskonformität 148

EK-01: Beachten von Gestaltungs-Best Practices 148

A.12 Gute Lesbarkeit 149

LE-01: Lesbarer Text 149
LE-02: Verständlicher Text 150

A.13 Hilfe anbieten 151

HI-01: Vorliegen einer Hilfestellung 151
HI-02: Komplexe Funktionen mit Hilfe begleiten 152
HI-03: Hilfestellung überspringen 153

Anhang A.1 Kontrolle und Einschränkung

<table>
<tr><th>Nr. KO-01</th><th colspan="3">Anforderungsmuster: Explizite Aktionskontrolle</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Die Applikation führt nur Funktionen aus, die der Nutzer ausgelöst hat. Dadurch bekommt der Nutzer Kontrolle über die Applikation und kann deren Funktionsweise lernen.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Control (Lidwell, Holden & Butler 2003)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2"></td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">KO-03: Funktionen pausieren und abbrechen</td></tr>
<tr><td>Konflikte</td><td colspan="2">EF-02: Körperlichen Einsatz minimieren,
FE-06: Automatisches Speichern überschriebener Inhalte</td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Die Applikation soll nur die Funktionen ausführen, die der Nutzer mit seinen Aktionen ausgelöst hat.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2">Die Applikation soll nur <Funktionen> ausführen, die der Nutzer mit <Aktionen> ausgelöst hat.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Funktionen></td><td>• vom Nutzer ausgelöste Funktionen</td></tr>
<tr><td><Aktionen></td><td>• Aktivieren von Bedienelementen
• Tätigen von Eingaben</td></tr>
<tr><td>Hinweis</td><td colspan="2">Diese Anforderung gilt nicht für Funktionen, durch deren automatisches Ausführen im Hintergrund ein Effektivitäts- oder Bequemlichkeitsnutzen entsteht.</td></tr>
</table>

<table>
<tr><th>Nr. KO-02</th><th colspan="3">Anforderungsmuster: Steuerbarkeit durch Vorliegen einer Navigation</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Der Nutzer kann sich auf der Bedienoberfläche vor- und zurückbewegen und erhält damit eine Navigationsmöglichkeit.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Steuerbarkeit (ISO Norm 9241)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">FE-04: Isoliertes Korrigieren von falschen Eingaben,
AI-03: Ständig erreichbare Navigation</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">KO-03: Funktionen pausieren und abbrechen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="4">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Die Applikation soll dem Nutzer ermöglichen, sich innerhalb der Dialogabfolge vor- und zurückzubewegen.</td></tr>
<tr><td rowspan="3">Erweiterung</td><td colspan="2">Die Applikation soll dem Nutzer ermöglichen, sich <Dialogabfolge> vor- und zurückzubewegen.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Dialogabfolge></td><td>• Gesamtheit der Dialoge der Applikation, in denen der Nutzer mit der Applikation interagieren kann</td></tr>
<tr><td></td><td>Hinweis</td><td colspan="2">Die Navigation der Applikation soll ein Bewegen in jede vorhandene Richtung ermöglichen. Das Zurückgehen kann auch durch ein Abbrechen ermöglicht werden.</td></tr>
</table>

<table>
<tr><th>Nr. KO-03</th><th colspan="3">Anforderungsmuster: Funktionen pausieren und abbrechen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer werden in die Lage versetzt, Dialog- und Funktionsabläufe zu steuern und zu kontrollieren.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Nutzerkontrolle (Scapin & Bastien 1997)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">KO-02: Steuerbarkeit durch Vorliegen einer Navigation</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">KO-01: Explizite Aktionskontrolle,
KO-02: Steuerbarkeit durch Vorliegen einer Navigation,
FE-03: Aktionen rückgängig machen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung 1</td><td colspan="2">Während die Applikation eine Funktion ausführt, soll der Nutzer die Möglichkeit haben, die Funktion zu unterbrechen und sie später fortzusetzen.</td></tr>
<tr><td>Erweiterung 1</td><td colspan="2">Während die Applikation <Funktion> ausführt, soll der Nutzer die Möglichkeit haben, <Funktion> zu unterbrechen und sie später fortzusetzen.</td></tr>
<tr><td>Standardisierte Anforderung 2</td><td colspan="2">Während die Applikation eine Funktion ausführt, soll der Nutzer die Möglichkeit haben, die Funktion abzubrechen.</td></tr>
<tr><td>Erweiterung 2</td><td colspan="2">Während die Applikation <Funktion> ausführt, soll der Nutzer die Möglichkeit haben, <Funktion> abzubrechen.</td></tr>
<tr><td rowspan="2"></td><td>Parameter</td><td>Werte</td></tr>
<tr><td><Funktion></td><td>• vom Nutzer ausgelöste Funktionen
• Hintergrundfunktionen</td></tr>
<tr><td>Hinweis</td><td colspan="2">Das Unterbrechen von ausgelösten Funktionen ist nur dann möglich, wenn dieses Ausführen lange genug dauert, um unterbrochen werden zu können. Das Unterbrechen eignet sich deshalb v.a. für längere Vorgänge. Das Abbrechen einer Funktion soll sowohl dann möglich sein, wenn der Nutzer sich in einem Dialog befindet, als auch, wenn die Applikation selbstständig Vorgänge ausführt.</td></tr>
</table>

<table>
<tr><th>Nr. KO-04</th><th colspan="3">Anforderungsmuster: Informationen zu Vor-Einstellungen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Die Nutzer wissen, dass das System in einem Einstellungsmenü Vor-Einstellungen vorgenommen hat.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Control (Jordan 1998)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">KO-05: Vor-Einstellung abspeichern,
FE-06: Automatisches Speichern überschriebener Inhalte</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">FE-05: Überprüfen von Eingaben vor Auslösen der Funktion,
SI-01: Systemstatus anzeigen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Wenn die Applikation in einem Einstellungsmenü selbstständig Einstellungen vornimmt, soll die Applikation den Nutzer über diese Vor-Einstellung informieren.</td></tr>
<tr><td rowspan="5">Erweiterung</td><td colspan="2">Wenn die Applikation in <Einstellungsmenü> selbstständig <Einstellungen> vornimmt, soll die Applikation den Nutzer <Informationen>.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Einstellungsmenü></td><td>•Dialog, der Eigenschaften einer Funktion festlegt</td></tr>
<tr><td><Einstellungen></td><td>• Kombination aus Eingaben in einem <Einstellungsmenü></td></tr>
<tr><td><Informationen></td><td>• es liegen <Einstellungen> vor
• wie lauten <Einstellungen>?</td></tr>
<tr><td>Hinweis</td><td colspan="2">Die Information muss nicht im ersten Schritt alle einzelnen Einstellungen aufzeigen, sollte aber ein Hinweis sein, dass Vor-Einstellungen vorgenommen wurden.</td></tr>
</table>

<table>
<tr><th>Nr. KO-05</th><th colspan="3">Anforderungsmuster: Vor-Einstellung abspeichern</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Der Nutzer kann häufig vorzunehmende Einstellungen als Vor-Einstellung abspeichern und erhält dadurch einen Effizienz- und Bequemlichkeitsnutzen.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Control (Jordan 1998)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">KO-04: Informationen zu Vor-Einstellungen,
IN-01: Aktive Personalisierung anbieten</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">IN-01: Aktive Personalisierung anbieten</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Wenn ein Einstellungsmenü auszufüllen ist, soll die Applikation dem Nutzer die Möglichkeit bieten, Einstellungen abzuspeichern und wiederzuverwenden.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2">Wenn <Einstellungsmenü> auszufüllen ist, soll die Applikation dem Nutzer die Möglichkeit bieten, <Einstellungen> abzuspeichern und wiederzuverwenden.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Einstellungsmenü></td><td>•Dialog, der Eigenschaften einer Funktion festlegt</td></tr>
<tr><td><Einstellungen></td><td>• Kombination aus Eingaben in einem <Einstellungsmenü></td></tr>
<tr><td>Hinweis</td><td colspan="2">Je nach Art der Funktion kann es für den Nutzer hilfreich sein, mehrere Einstellungen abzuspeichern.</td></tr>
</table>

<table>
<tr><th>Nr. KO-06</th><th colspan="3">Anforderungsmuster: Zwei Wege zur Aufgabenbewältigung</th></tr>
<tr><td rowspan="6">Metadaten</td><td>Ziel</td><td colspan="2">Die Applikation bietet zwei Wege zur Aufgabenbewältigung an: einen einfachen Weg, mit wenigen Einstellungsmöglichkeiten und einen Weg für erfahrene Nutzer mit allen Einstellungsmöglichkeiten.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Control (Lidwell, Holden & Butler et al. 2003)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2"></td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">HI-02: Komplexe Funktionen mit Hilfe begleiten,
EF-05: Aufteilen komplexer Aufgaben,
IN-03: Mehrere Aktionen gleichzeitig ausführen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td colspan="3" style="display:none"></td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Wenn eine Funktion Einstellungen erlaubt, soll die Applikation dem Nutzer die Funktion einmal mit Vor-Einstellungen und einmal mit allen Einstellungsmöglichkeiten anbieten.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2">Wenn <Funktion> <Einstellungen> erlaubt, soll die Applikation dem Nutzer <Funktion> einmal mit Vor-<Einstellungen> und einmal mit allen <Einstellungen> anbieten.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Funktionen></td><td>• vom Nutzer ausgelöste Funktionen</td></tr>
<tr><td><Einstellungen></td><td>• Kombination aus Eingaben in einer Funktion</td></tr>
<tr><td>Hinweis</td><td colspan="2">Es ist zu prüfen, wie kompliziert Aufgaben sind. Das Anbieten von zwei Wegen bietet sich nur bei sehr komplexen Aufgaben oder Systemen an. Mehr als zwei Wege können die Bedienoberfläche überladen und kompliziert machen.</td></tr>
</table>

<table>
<tr><th>Nr. KO-07</th><th colspan="3">Anforderungsmuster: Kein Anbieten nicht verfügbarer Funktionen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Es werden nur Funktionen angezeigt, die aktuell ausführbar sind. Dadurch wird das Design verschlankt und die kognitive Belastung der Nutzer verringert.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Constraint (Lidwell, Holden & Butler et al. 2003)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2"></td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">AI-01: Wichtige Informationen hervorheben</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Ist eine Funktion nicht ausführbar, soll die Applikation die Funktion nicht anbieten.</td></tr>
<tr><td rowspan="3">Erweiterung</td><td colspan="2">Ist <Funktion> nicht ausführbar, soll die Applikation <Funktion> nicht anbieten.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Funktion></td><td>• vom Nutzer ausgelöste Funktion</td></tr>
<tr><td>Hinweis</td><td colspan="2">Nicht verfügbare Funktionen können entweder vollständig ausgeblendet werden oder werden so dargestellt, dass ihre Nicht-Verfügbarkeit für den Nutzer erkennbar wird.</td></tr>
</table>

Anhang A.2 Effizienz

Nr. EF-01	**Anforderungsmuster: Inhalt herausstellen**		
Metadaten	Ziel	Der Nutzer kann den Inhalt einer Applikation schnell erfassen und wird nicht von den Bedienelementen der Navigation abgelenkt.	
	Grundlage	Ablenkung (Blair-Early & Zender 2008)	
	Abhängigkeiten	AI-01: Wichtige Informationen hervorheben	
	Verknüpfungen	AI-01: Wichtige Informationen hervorheben	
	Konflikte		
Vorlage	Standardisierte Anforderung	Hat eine Funktion den Zweck, Inhalt darzustellen, sollen die Bedienelemente nicht von diesem Inhalt ablenken.	
	Erweiterung	Zeigt <Funktion> <Inhalt>, sollen <Bedienelemente> nicht von <Inhalt> ablenken.	
		Parameter	Werte
		<Funktion>	• ausgewählte Funktionalität der Applikation
		<Inhalt>	• von der Applikation dargestellte Objekte, die nicht ausschließlich zur Navigation gehören
		<Bedienelemente>	• von der Applikation dargestellte Objekte, die zur Navigation dienen
	Hinweis	Das Verhältnis von Inhalt zu Navigation muss in Zusammenhang mit der Bildschirmgröße gesehen werden.	

<table>
<tr><th>Nr. EF-02</th><th colspan="3">Anforderungsmuster: Aufwand des Nutzers minimieren</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Der Nutzer führt nur die notwendigsten Handlungen durch und erlebt damit eine effiziente und bequeme Nutzung.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Effizienz (Knittle, Ruth & Gardner 1987)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">EF-03: Körperlichen Einsatz minimieren,
EF-04: Verständliche Anweisungen</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">EF-03: Körperlichen Einsatz minimieren,
EF-04: Verständliche Anweisungen,
IN-03: Mehrere Aktionen gleichzeitig ausführen</td></tr>
<tr><td>Konflikte</td><td colspan="2">KO-01:</td></tr>
<tr><td rowspan="5">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Der Nutzer soll nur Aktionen ausführen müssen, die seine Entscheidung erfordern.</td></tr>
<tr><td rowspan="3">Erweiterung</td><td colspan="2">Der Nutzer soll nur <Aktionen> ausführen müssen, die seine Entscheidung erfordern.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Aktionen></td><td>• Aktivieren von Bedienelementen
• Tätigen von Eingaben</td></tr>
<tr><td>Hinweis</td><td colspan="2">Die Applikation kann alle Prozesse, die keine Entscheidung des Nutzers erfordern, automatisiert ausführen.</td></tr>
</table>

<table>
<tr><th>Nr. EF-03</th><th colspan="3">Anforderungsmuster: Körperlichen Einsatz minimieren</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Der Nutzer führt nur die notwendigsten Bewegungen durch und erlebt damit eine effiziente und bequeme Nutzung.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Effizienz (Knittle, Ruth & Gardner 1987)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">EF-02: Aufwand des Nutzers minimieren,
EF-04: Verständliche Anweisungen</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">EF-02: Aufwand des Nutzers minimieren,
EF-04: Verständliche Anweisungen,
IN-03: Mehrere Aktionen gleichzeitig ausführen</td></tr>
<tr><td>Konflikte</td><td colspan="2">FE-01: Bestätigung kritischer Aktionen</td></tr>
<tr><td rowspan="4">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Das Durchführen von Aktionen durch den Nutzer soll mit dem geringstmöglichen körperlichen Einsatz erfolgen.</td></tr>
<tr><td rowspan="3">Erweiterung</td><td colspan="2">Das Durchführen von <Aktion> soll mit dem geringstmöglichen körperlichen Einsatz erfolgen.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Aktion></td><td>• Aktivieren von Bedienelementen
• Tätigen von Eingaben</td></tr>
<tr><td></td><td>Hinweis</td><td colspan="2">Der Nutzer soll keine Zwischenschritte ausführen müssen, die zusammengefasst oder übersprungen werden können. Zur Bedienung sollen - je nach Art und Beschaffenheit des Endgerätes - möglichst einfache, kurze Bewegungen benutzt werden.</td></tr>
</table>

<table>
<tr><th>Nr. EF-04</th><th colspan="3">Anforderungsmuster: Verständliche Anweisungen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Der Nutzer versteht und lernt Aktionen mit Hilfe verständlicher Bilder und Textanweisungen</td></tr>
<tr><td>Grundlage</td><td colspan="2">Geringe kognitive Belastung (Hix & Hartson 1993)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">HI-01: Vorliegen einer Hilfestellung,
HI-02: Komplexe Funktionen mit Hilfe begleiten,
LE-01: Lesbarer Text,
LE-02: Verständlicher Text</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">EF-02: Aufwand des Nutzers minimieren,
EF-03: Körperlichen Einsatz minimieren,
HI-01: Vorliegen einer Hilfestellung,
HI-02: Komplexe Funktionen mit Hilfe begleiten,
LE-01: Lesbarer Text,
LE-02: Verständlicher Text</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="9">Vorlage</td><td>Standardisierte Anforderung 1</td><td colspan="2">Wenn der Nutzer eine oder mehrere Aktionen durchführen soll, soll die Applikation Bildanweisungen mit klaren, unmissverständlichen Bildern und Symbolen verwenden.</td></tr>
<tr><td>Erweiterung 1</td><td colspan="2">Wenn der Nutzer <Aktionen> durchführen soll, soll die Applikation <Bildanweisungen> mit klaren, unmissverständlichen Bildern und Symbolen verwenden.</td></tr>
<tr><td>Standardisierte Anforderung 2</td><td colspan="2">Wenn der Nutzer eine oder mehrere Aktionen durchführen soll, soll die Applikation Textanweisungen mit kurzen, klaren Sätzen verwenden.</td></tr>
<tr><td>Erweiterung 2</td><td colspan="2">Wenn der Nutzer <Aktionen> durchführen soll, soll die Applikation <Textanweisungen> mit kurzen, klaren Sätzen verwenden.</td></tr>
<tr><td rowspan="4"></td><td>Parameter</td><td>Werte</td></tr>
<tr><td><Aktionen></td><td>• Aktivieren von Bedienelementen
• Tätigen von Eingaben</td></tr>
<tr><td><Bildanweisungen></td><td>• Hilfestellung in Form von Bildern oder Symbolen</td></tr>
<tr><td><Textanweisungen></td><td>• Hilfestellug in Form von Text</td></tr>
<tr><td>Hinweis</td><td colspan="2">Es sind Bilder oder Symbole zu finden, die dem Nutzer auf einfachste Weise erklären, was zu tun ist. Dazu empfehlen sich Motive, die eine universelle Bedeutung haben. Der in Textanweisungen verwendete Text sollte kurze, einfache Satzstellungen aufweisen, aktiv formuliert sein und keine Füllwörter enthalten. Ob Bilder und Texte im Kontext der Applikation tatsächlich verstanden werden, lässt sich mit einem Nutzertest herausfinden.</td></tr>
</table>

<table>
<tr><th>Nr. EF-05</th><th colspan="3">Anforderungsmuster: Aufteilen komplexer Aufgaben</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Komplexe Aufgaben, die den Nutzer überfordern könnten, werden in Teil-Schritte aufgeteilt. Dadurch wird die Aufgabe für den Nutzer überschaubarer und verständlicher.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Effizienz (Hix & Hartson 1993)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">EF-07: Kein Merken-Müssen von Informationen</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">KO-06: Zwei Wege der Aufgabenbewältigung</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Die für die Durchführung der Aufgabe notwendige Aktionsabfolge soll in mehrere Teilschritte unterteilt werden.</td></tr>
<tr><td rowspan="5">Erweiterung</td><td colspan="2">Die für die Durchführung von <Aufgabe> notwendige <Aktionen> soll in <Teilschritte> unterteilt werden.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Aufgabe></td><td>• Gesamtheit an <Aktionen> zur Erreichung eines Ziels</td></tr>
<tr><td><Aktionen></td><td>• Aktivieren von Bedienelementen
• Tätigen von Eingaben</td></tr>
<tr><td><Teilschritte></td><td>• Mehrere geordnete Bündel aus Aktionen einer Aufgabe</td></tr>
<tr><td>Hinweis</td><td colspan="2">Die Teilschritte sind so zu gestalten, dass sie die Abfolge der durchzuführenden Aktionen vereinfachen ohne sie dabei unnötig zu überladen. Das Aufteilen in Teilschritte empfiehlt sich bei sehr komplexen und umfangreichen Aufgaben. Eine Aufgabe ist als komplex zu betrachten, wenn Erstnutzer bei der Aufgabenerfüllung überfordert sind oder Fehler machen.</td></tr>
</table>

<table>
<tr><th>Nr. EF-06</th><th colspan="3">Anforderungsmuster: Übersichtlichkeit</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Der Nutzer bekommt nur die Dinge zu sehen, die für die aktuelle Aufgabe wichtig sind. Dadurch wird er weniger stark abgelenkt, macht weniger Fehler und führt Aktionen effizienter durch.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Geringe kognitive Belastung (Knittle, Ruth & Gardner 1987)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">EF-07: Kein Merken-Müssen von Informationen,
SI-01: Systemstatus anzeigen,
AI-01: Wichtige Informationen hervorheben</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">AI-01: Wichtige Informationen hervorheben</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Wenn der Nutzer eine oder mehrere Aktionen durchführen soll, soll die Applikation nur Informationen anzeigen, die zur aktuell ausgewählten Funktion gehören.</td></tr>
<tr><td rowspan="5">Erweiterung</td><td colspan="2">Wenn der Nutzer <Aktionen> durchführen soll, soll die Applikation nur <Informationen>, <Funktion>.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Aktionen></td><td>• Aktivieren von Bedienelementen
• Tätigen von Eingaben</td></tr>
<tr><td><Informationen></td><td>• alle denkbaren zu <Funktion> gehörenden Informationen</td></tr>
<tr><td><Funktion></td><td>• von Nutzer ausgelöste Funktion</td></tr>
<tr><td>Hinweis</td><td colspan="2">Bei der Auswahl der angezeigten Informationen ist zu prüfen, ob und inwieweit die Informationen für die aktuelle Aktion notwendig sind.</td></tr>
</table>

<table>
<tr><td>Nr. EF-07</td><td colspan="3">Anforderungsmuster: Kein Merken-Müssen von Informationen</td></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Die kognitive Belastung des Nutzers wird verringert, indem er sich während der Benutzung keine Informationen merken muss.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Geringe kognitive Belastung (Knittle, Ruth & Gardner 1987)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">EF-06: Übersichtlichkeit</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2"></td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Wenn eine Funktion mit mehreren Aktionen ausgeführt wird, soll die Applikation dem Nutzer stets alle notwendigen Informationen anzeigen.</td></tr>
<tr><td rowspan="5">Erweiterung</td><td colspan="2">Wenn <Funktion> mit mehreren <Aktionen> ausgeführt wird, soll die Applikation dem Nutzer stets <Informationen> anzeigen.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Funktion></td><td>• vom Nutzer ausgeführte Funktionen</td></tr>
<tr><td><Aktionen></td><td>• Aktivieren von Bedienelementen
• Tätigen von Eingaben</td></tr>
<tr><td><Informationen></td><td>• jede Information, die zum Ausführen von <Funktion> notwendig ist</td></tr>
<tr><td>Hinweis</td><td colspan="2">Die Applikation sollte bei jeder auszuführenden Aktion die dafür notwendigen Informationen anzeigen, v. a. jene, die in den Arbeitsschritten zuvor erarbeitet wurden. Dies gilt v. a., wenn sich eine Aufgabe in mehrere Teilschritte aufteilt.</td></tr>
</table>

Anhang A.3 Individualisierbarkeit

<table>
<tr><th>Nr. IN-01</th><th colspan="2">Anforderungsmuster: Aktive Personalisierung anbieten</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td>Nutzer können Dialoge und Funktionen aktiv an ihre Bedürfnisse und Vorlieben anpassen.</td></tr>
<tr><td>Grundlage</td><td>Individualisierbarkeit (Rampl 2007)</td></tr>
<tr><td>Abhängigkeiten</td><td>KO-04: Informationen zu Vor-Einstellung,
KO-05: Vor-Einstellung abspeichern</td></tr>
<tr><td>Verknüpfungen</td><td>IN-02: Passive Personalisierung anbieten,
KO-05: Vor-Einstellung abspeichern</td></tr>
<tr><td>Konflikte</td><td></td></tr>
<tr><td rowspan="3">Vorlage</td><td>Standardisierte Anforderung</td><td>Die Applikation soll dem Nutzer erlauben, Einstellungen an einer Funktion vorzunehmen.</td></tr>
<tr><td>Erweiterung</td><td>Die Applikation soll dem Nutzer erlauben, <Einstellungen> an <Funktion> vorzunehmen.<table><tr><td>Parameter</td><td>Werte</td></tr><tr><td><Funktion></td><td>• vom Nutzer ausgelöste Funktion</td></tr><tr><td><Einstellungen></td><td>• Eingabeoptionen
• Darstellungsoptionen
• Sicherheitseinstellungen
• sonstige Vor-Einstellungen</td></tr></table></td></tr>
<tr><td>Hinweis</td><td>Es ist zu analysieren, an welchen Stellen einer Funktion Nutzer sich persönliche Einstellungen wünschen. Die Einstellungen können lediglich das Äußere der Funktion betreffen oder deren Funktionalität.</td></tr>
</table>

<table>
<tr><th>Nr. IN-02</th><th colspan="3">Anforderungsmuster: Passive Personalisierung anbieten</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Das System erfasst die Art und Weise, mit der ein Nutzer eine Funktion nutzt, und blendet Optionen aus, die nie verwendet wurden. Dadurch wird die Funktion geradliniger und ermöglicht dem Nutzer eine effizientere Nutzung.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Individualisierbarkeit (Rampl 2007)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">KO-04: Informationen zu Vor-Einstellung,
KO-05: Vor-Einstellung abspeichern</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">IN-01: Aktive Personalisierung anbieten</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Die Applikation soll dem Nutzer nach einer bestimmten Zahl an Nutzungen von einer Funktion die Möglichkeit bieten, nicht verwendete Optionen auszublenden.</td></tr>
<tr><td rowspan="5">Erweiterung</td><td colspan="2">Die Applikation soll dem Nutzer nach <X> Nutzungen von <Funktion> die Möglichkeit bieten, nicht verwendete <Optionen> auszublenden.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Funktion></td><td>• von Nutzer ausgelöste Funktion</td></tr>
<tr><td><X></td><td>• festgelegte Anzahl an Benutzungen</td></tr>
<tr><td><Optionen></td><td>• (Teil-)Funktionen
• Einstellungen
• Bedienelemente</td></tr>
<tr><td>Hinweis</td><td colspan="2">Um das Ausblenden nicht-verwendeter Optionen anzubieten, muss das Nutzerverhalten analysiert werden.</td></tr>
</table>

<table>
<tr><th>Nr. IN-03</th><th colspan="3">Anforderungsmuster: Mehrere Aktionen gleichzeitig ausführen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Erfahrene Nutzer können mehrere Aktionen gleichzeitig durchführen, sparen dadurch Zeit und Mühen und erhalten eine effizientere Form der Nutzung</td></tr>
<tr><td>Grundlage</td><td colspan="2">Abkürzungen (Scapin & Bastien 1997)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2"></td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">EF-02: Aufwand des Nutzers minimieren,
EF-03: Körperlichen Einsatz minimieren,
IN-03: Mehrere Aktionen gleichzeitig ausführen,
HI-03: Hilfestellung überspringen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Die Applikation soll das gleichzeitige Ausführen mehrerer Aktionen durch eine einzige Aktion ermöglichen.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2">Die Applikation soll das gleichzeitige Ausführen von <Aktionen> durch <Aktion> ermöglichen.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Aktionen></td><td>• Aktivieren von Bedienelementen

• Tätigen von Eingaben</td></tr>
<tr><td><Aktion></td><td>• einzelne Eingabe zum Auslösen mehrer <Aktionen></td></tr>
<tr><td>Hinweis</td><td colspan="2">Das gleichzeitige Ausführen mehrerer Aktionen kann über Vorlagen gelöst werden, die entweder von der Applikation angeboten oder vom Nutzer angelegt werden.</td></tr>
</table>

Anhang A.4 Fehler-Handhabung

<table>
<tr><th>Nr. FE-01</th><th colspan="3">Anforderungsmuster: Bestätigung kritischer Funktionen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer werden auf die Folgen kritischer Aktionen hingewiesen und benutzen diese nicht versehentlich.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Confirmation (Lidwell, Holden & Butler 2003)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2"></td></tr>
<tr><td>Verknüpfungen</td><td colspan="2"></td></tr>
<tr><td>Konflikte</td><td colspan="2">EF-03: Körperlichen Einsatz minimieren</td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Löst der Nutzer eine kritische Funktion aus, soll die Applikation einen Dialog öffnen, in dem der Nutzer die Funktion ein zweites Mal bestätigen oder abbrechen kann.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2">Löst der Nutzer <Funktion> aus, soll die Applikation <Dialog> öffnen, in dem der Nutzer <Funktion> ein zweites mal bestätigen oder abbrechen kann.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Funktion></td><td>• Auslösen einer kritischen Funktion</td></tr>
<tr><td><Dialog></td><td>• hervorgestellter Dialog</td></tr>
<tr><td>Hinweis</td><td colspan="2">Das Bestätigen von Funktionen verlangsamt die Benutzung der Applikation und sollte daher nur bei Funktionen eingesetzt werden, deren Folgen nur schwer oder gar nicht rückgängig gemacht werden können. Der sich öffnende Dialog kann auf die Folgen der Funktion hinweisen.</td></tr>
</table>

<table>
<tr><th>Nr. FE-02</th><th colspan="3">Anforderungsmuster: Fehler-Benachrichtigungen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer sehen sofort, dass sie einen Fehler gemacht haben und erkennen, wie dieser zustande kam. Sie erfahren außerdem, wie die Fehler korrigiert oder rückgängig gemacht werden können.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Error prevention and recovery (Jordan 1998)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">FE-03: Aktionen rückgängig machen,
RM-01: Rückmeldung bei Aktionen</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">RM-04: Erfolgreiche Bearbeitung anzeigen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="9">Vorlage</td><td>Standardisierte Anforderung 1</td><td colspan="2">Macht der Nutzer einen Fehler, soll die Applikation ihn unmittelbar darauf hinweisen.</td></tr>
<tr><td>Erweiterung 1</td><td colspan="2">Macht der Nutzer < Fehler>, soll die Applikation ihn unmittelbar <Information>.</td></tr>
<tr><td>Standardisierte Anforderung 2</td><td colspan="2">Zeigt die Applikation eine Fehler-Benachrichtigung an, soll diese explizite Hinweise zur Problemlösung enthalten.</td></tr>
<tr><td>Erweiterung 2</td><td colspan="2">Zeigt die Applikation <Fehler-Benachrichtigung>, soll <Fehler-Benachrichtigung> <Information>.</td></tr>
<tr><td rowspan="4"></td><td>Parameter</td><td>Werte</td></tr>
<tr><td><Fehler></td><td>• Eingabe ungültiger Werte
• ungültiges Auslösen eines Bedienelementes</td></tr>
<tr><td><Information></td><td>• Art und Quelle des Fehlers
• genaue Position des Fehlers
• Art und Quelle des Fehlers
• Hinweis zur behebung des Fehlers</td></tr>
<tr><td><Fehler-Benachrichtigung></td><td>• <Information> über Fehler</td></tr>
<tr><td>Hinweis</td><td colspan="2">Die Fehler-Benachrichtigung erscheint idealerweise noch während der Nutzer eine Eingabe macht und genau an der Stelle der jeweiligen Eingabe. Die Hinweise der Fehlermeldung sollten sachlich und positiv formuliert sein, die Applikation sollte die Schuld auf sich nehmen und den Nutzer ermutigen, das Richtige zu tun.</td></tr>
</table>

<table>
<tr><th>Nr. FE-03</th><th colspan="3">Anforderungsmuster: Aktionen rückgängig machen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer können Aktionen und Eingaben rückgängig machen, ohne den aktuellen Dialog/die aktuelle Funktion zu verlassen. So können mühelos Fehler korrigiert werden.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Forgiveness (Blair-Early & Zender 2008)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">FE-02: Fehler-Benachrichtigung,
KO-02: Steuerbarkeit durch Vorliegen einer Navigation,
KO-03: Funktionen pausieren und abbrechen,
FE-05: Überprüfen von Eingaben vor Auslösen der Funktion</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">KO-03: Funktionen pausieren und abbrechen,
FE-05: Überprüfen von Eingaben vor Auslösen der Funktion,
RM-04: Erfolgreiche Bearbeitung anzeigen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Der Nutzer soll die Möglichkeit haben, Aktionen unmittelbar rückgängig zu machen.</td></tr>
<tr><td rowspan="3">Erweiterung</td><td colspan="2">Der Nutzer soll die Möglichkeit haben, <Aktionen> unmittelbar rückgängig zu machen.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Aktionen></td><td>• Aktivieren von Bedienelementen

• Tätigen von Eingaben</td></tr>
<tr><td>Hinweis</td><td colspan="2">Unmittelbares Rückgängig-machen bedeutet, dass Eingaben entweder während der Eingabe oder direkt danach entfernt oder ausgetauscht werden können.</td></tr>
</table>

Nr. FE-04	**Anforderungsmuster: Isoliertes Korrigieren von falschen Eingaben**		
Metadaten	Ziel	Nutzer können falsche Eingaben korrigieren, ohne richtige Eingaben erneut eingeben zu müssen. Dies spart Zeit und ermöglicht ein bequemeres, effizienteres Nutzen einer Funktion.	
	Grundlage	Forgiveness (Blair-Early & Zender 2008)	
	Abhängigkeiten	FE-02: Fehler-Benachrichtigung, FE-03: Aktionen rückgängig machen	
	Verknüpfungen	FE-03: Aktionen rückgängig machen, FE-04: Isoliertes Korrigieren von falschen Eingaben	
	Konflikte		
Vorlage	Standardisierte Anforderung	Macht der Nutzer Fehler, soll er die falschen Eingaben unmittelbar korrigieren können, ohne die richtigen Eingaben neu eingeben zu müssen.	
	Erweiterung	Macht der Nutzer <Fehler>, soll er falsche <Eingabe> unmittelbar korrigieren können, ohne die richtigen <Eingaben> neu eingeben zu müssen.	
		Parameter	Werte
		<Fehler>	• Eingabe ungültiger <Eingaben>
		<Eingaben>	• Zeichen • Werte • Bewegungen • Geräusche
	Hinweis	Das isolierte Korrigieren von Falscheingaben ist v.a. bei längeren/umfangreicheren Dialogen mit vielen Eingabefeldern sinnvoll und sollte in Kombination mit punktgenauen Fehler-Benachrichtigungen eingesetzt werden.	

<table>
<tr><th>Nr. FE-05</th><th colspan="3">Anforderungsmuster: Überprüfen von Eingaben vor Auslösen der Funktion</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer können getätigte Eingaben überprüfen, bevor sie eine Funktion auslösen</td></tr>
<tr><td>Grundlage</td><td colspan="2">Forgiveness (Blair-Early & Zender 2008)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">FE-04: Isoliertes Korrigieren von falschen Eingaben,
KO-04: Informationen zu Vor-Einstellung</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">FE-03: Aktionen rückgängig machen,
FE-04: Isoliertes Korrigieren von falschen Eingaben,
KO-04: Informationen zu Vor-Einstellung</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Der Nutzer soll die Möglichkeit haben, alle Eingaben eines Eingabeformulars zu überprüfen, bevor er die dazugehörige Funktion auslöst.</td></tr>
<tr><td rowspan="5">Erweiterung</td><td colspan="2">Der Nutzer soll die Möglichkeit haben, alle <Eingaben> in <Eingabeformular> zu überprüfen bevor er <Funktion> auslöst.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Eingaben></td><td>• Zeichen
• Werte
• Bewegungen
• Geräusche</td></tr>
<tr><td><Eingabeformular></td><td>• Kombination notwendiger Eingaben zum Ausführen von <Funktion></td></tr>
<tr><td><Funktion></td><td>• von Nutzer ausgelöste Funktion</td></tr>
<tr><td>Hinweis</td><td colspan="2">Das Überprüfen von Eingaben eignet sich für kritische Funktionen oder für Funktionen, für die sehr umfangreiche/fehleranfällige Eingaben notwendig sind.</td></tr>
</table>

Nr. FE-06	**Anforderungsmuster: Automatisches Speichern überschriebener Inhalte**		
Metadaten	Ziel	Die Applikation speichert Inhalte, die der Nutzer erstellt hat, nachdem sie überschrieben wurden. Macht der Nutzer bei seiner Eingabe Fehler, stehen ihm die vorherigen Eingaben zur Verfügung.	
	Grundlage	Forgiveness (Blair-Early & Zender 2008)	
	Abhängigkeiten		
	Verknüpfungen		
	Konflikte	KO-01: Explizite Aktionskontrolle	
Vorlage	Standardisierte Anforderung	Wenn der Nutzer einmal erstellte Inhalte ändert, soll die Applikation die vorherigen Inhalte ohne Aufforderung des Nutzers speichern und abrufbar machen.	
	Erweiterung	Wenn der Nutzer <Inhalte> ändert, soll die Applikation die vorherigen <Inhalte> ohne Aufforderung des Nutzers speichern und abrufbar machen.	
		Parameter	Werte
		<Inhalte>	• alle denkbaren Informationen, die durch das Zusammenwirken von Nutzer und Applikation erstellt wurden.
	Hinweis	Die gespeicherten Inhalte müssen nicht prominent angezeigt werden, der Nutzer sollte bei Bedarf aber erkennen, wo er sie findet.	

Anhang A.5 Eintrittspunkte

<table>
<tr><th>Nr. EP-01</th><th colspan="3">Anforderungsmuster: Startpunkt anbieten</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Die Applikation erhält einen Startpunkt, von dem aus die Navigation für den Nutzer beginnt. Dies hilft dem Nutzer, die Benutzung zu erlernen.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Entry Point (Lidwell, Holden & Butler 2003)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">KO-02: Steuerbarkeit durch Vorliegen einer Navigation,
EP-02: Jederzeitiges Zurückkehren zum Startpunkt,
EP-03: Keine Barrieren im Startpunkt</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">WE-01: Orientierung anbieten</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Die Navigation soll einen offensichtlichen Startpunkt haben.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2"><Navigation> soll einen offensichtlichen <Startpunkt> haben.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Navigation></td><td>• Gesamtheit der Bedienelemente der Applikation
• Gesamtheit der Bedienelemente einer Funktion</td></tr>
<tr><td><Startpunkt></td><td>• Ausgangspunkt für <Navigation>
• Ziel bei Abbrechen von Funktionen</td></tr>
<tr><td>Hinweis</td><td colspan="2">Der Startpunkt sollte so konstruiert sein, dass Nutzer ihn immer dann erreichen, wenn sie die Applikation/Funktion aufrufen oder einen Vorgang abbrechen.</td></tr>
</table>

<table>
<tr><th>Nr. EP-02</th><th colspan="3">Anforderungsmuster: Jederzeitiges Zurückkehren zum Startpunkt</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer können Dialoge und Funktionen jederzeit abbrechen, indem sie zum Startpunkt der Navigation zurückkehren.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Obvious Start (Blair-Early-Ender 2008)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">EP-01: Startpunkt anbieten</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2"></td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Der Nutzer soll jederzeit die Möglichkeit haben, zum Startpunkt zurückzukehren.</td></tr>
<tr><td rowspan="3">Erweiterung</td><td colspan="2">Der Nutzer soll jederzeit die Möglichkeit haben, zu <Startpunkt> zurückzukehren.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Startpunkt></td><td>• Ausgangspunkt für Navigation der Applikation
• Ausgangspunkt der Navigation einer Funktion</td></tr>
<tr><td>Hinweis</td><td colspan="2">Die jederzeitige Möglichkeit, zum Startpunkt zurückzukehren, wird idealerweise durch ein permanent sichtbares Bedienelement oder eine permanent verfügbare Funktion ermöglicht.</td></tr>
</table>

<table>
<tr><th>Nr. EP-03</th><th colspan="3">Anforderungsmuster: Keine Barrieren im Startpunkt</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Der Startpunkt einer Navigation enthält keinerlei Barrieren und ermöglicht Nutzern das ungehinderte Nutzen der Applikation.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Start Screens (Cappel & Huang 2007)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">EP-01: Startpunkt anbieten</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2"></td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Der Startpunkt der Applikation soll das unmittelbare Verwenden ihrer Funktionen erlauben.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2"><Startpunkt> der Applikation soll das unmittelbare Verwenden <Funktionen> erlauben.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Startpunkt></td><td>• Ausgangspunkt für Navigation der Applikation
• Ausgangspunkt der Navigation einer Funktion</td></tr>
<tr><td><Funktionen></td><td>• von Nutzer auslösbare Funktionen
• sonstige Funktionen & Inhalte</td></tr>
<tr><td>Hinweis</td><td colspan="2">Auch scheinbar einladende Startbildschirme, die dem Startpunkt vorgeschaltet sind, verzögern die Benutzung der Applikation und machen diese ineffizient. Der Startpunkt darf Barrieren enthalten, wenn Inhalte der Applikation nicht frei zugänglich sind.</td></tr>
</table>

Nr. EP-04	**Anforderungsmuster: Überblick-Funktion des Startpunkts**		
Metadaten	Ziel	Der Startpunkt der Applikation verdeutlicht dem Nutzer deren Kernelemente wie Hauptfunktionen und Informationsbereiche.	
	Grundlage	Clear Overview of the Service Entity (Kaasinen 2005)	
	Abhängigkeiten	EP-01: Startpunkt anbieten, EP-03: Keine Barrieren im Startpunkt, AI-01: Wichtige Informationen hervorheben	
	Verknüpfungen	EP-03: Keine Barrieren im Startpunkt, WE-01: Orientierung anbieten, AI-01: Wichtige Informationen hervorheben	
	Konflikte		
Vorlage	Standardisierte Anforderung	Der Startpunkt der Navigation soll einen Überblick über die Funktionen der Applikation enthalten.	
	Erweiterung	<Startpunkt> von <Navigation> soll einen Überblick über <Funktionen> der Applikation enthalten.	
		Parameter	Werte
		<Startpunkt>	• Ausgangspunkt für <Navigation>
		<Navigation>	• Gesamtheit der Bedienelemente der Applikation
		<Funktionen>	• vom Nutzer auslösbare Funktionen
	Hinweis	Es sollten nicht zu viele Informationen angezeigt werden, die nicht zum eigentlichen Service gehören. Bei der Gestaltung des Startpunkts und dessen Überblick sollten die Bildschirmgrößen der Endgeräte der Nutzer berücksichtigt werden. Der Startpunkt sollte die Aufmerksamkeit der Nutzer auf die Funktionen der Applikation lenken und zu deren Benutzung einladen.	

Anhang A.6 Wegfindung

Nr. WE-01	Anforderungsmuster: Orientierung anbieten	
Metadaten	Ziel	Der Nutzer weiß zu jedem Zeitpunkt, wo er sich innerhalb der Navigation befindet.
	Grundlage	Orientation (Lidwell, Holden & Butler 2003)
	Abhängigkeiten	EP-01: Startpunkt anbieten, EP-02: Jederzeitiges Zurückkehren zum Startpunkt, EP-04: Überblick-Funktion des Startpunkts, WE-02: Aussagekräftige Bedienelemente
	Verknüpfungen	EP-01: Startpunkt anbieten, EP-04: Überblick-Funktion des Startpunkts
	Konflikte	
Vorlage	Standardisierte Anforderung	Die Applikation soll dem Nutzer zu jedem Zeitpunkt anzeigen, an welcher Stelle der Navigation er sich befindet.
	Hinweis	Um dem Nutzer mitzuteilen, wo er sich befindet, können Orientierungspunkte oder der bereits zurückgelegte Weg angezeigt werden. Alternativ kann jede aufrufbare Seite einen Titel erhalten.

Nr. WE-02	Anforderungsmuster: Aussagekräftige Bedienelemente	
Metadaten	Ziel	Nutzer erkennen, wohin sie gehen werden, wenn sie ein Bedienelement benutzen.
	Grundlage	Selbstbeschreibungsfähigkeit (Rampl 2007)
	Abhängigkeiten	WE-01: Orientierung anbieten, CO-02: Konsistent gestaltete Funktionen, CO-03: Konsistent gestaltete Informationen
	Verknüpfungen	CO-02: Konsistent gestaltete Funktionen, CO-03: Konsistent gestaltete Informationen
	Konflikte	
Vorlage	Standardisierte Anforderung	Die Navigation der Applikation soll aussagekräftige und klar benannte Bedienelemente haben.
	Hinweis	Die Sprache der Bedienelemente sollte kurz, prägnant und in der ganzen Applikation konsistent sein.

Anhang A.7 Anordnung von Informationen

<table>
<tr><th>Nr. AI-01</th><th colspan="3">Anforderungsmuster: Wichtige Informationen hervorheben</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer erkennen schneller die Kernaspekte der Applikation oder einer Funktion. Sie können sich leichter orientieren und die Applikation verstehen.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Hierarchy (Lidwell, Holden & Butler 2003)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">EF-01: Inhalt herausstellen</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">EP-04: Überblick-Funktion des Startpunkts,
EF-01: Inhalt herausstellen,
EF-06: Übersichtlichkeit,
KO-07: Kein Anbieten nicht verfügbarer Funktionen,
AI-01: Wichtige Informationen hervorheben</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Die Applikation soll wichtige Informationen gegenüber weniger wichtigen Informationen hervorgehoben darstellen.</td></tr>
<tr><td rowspan="3">Erweiterung</td><td colspan="2">Die Applikation soll <Informationen> gegenüber <Informationen> hervorgehoben darstellen.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Informationen></td><td>• Inhalt (Text, Bilder, Bewegungen, Geräusche)
• Bedienelemente aller Art
• sonstige Bestandteile der Bedienoberfläche</td></tr>
<tr><td>Hinweis</td><td colspan="2">Alle Informationen und Bestandteile der Applikation sollten ihrer Wichtigkeit entsprechend geordnet und angezeigt werden. Bei komplexeren Informationsstrukturen empfiehlt sich das Anlegen und Durchhalten einer Hierarchie.</td></tr>
</table>

<table>
<tr><th>Nr. AI-02</th><th colspan="3">Anforderungsmuster: Informationen gruppieren</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Die Komplexität von dargestellten Informationen wird verringert, indem Informationen sinnvoll gruppiert werden. Nutzer können Zusammenhänge in der Applikation schneller erfassen und lernen.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Layering (Lidwell, Holden & Butler 2003)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">AI-01: Wichtige Informationen hervorheben</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">AI-01: Wichtige Informationen hervorheben</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Die Applikation soll Informationen in der Darstellung gruppieren.</td></tr>
<tr><td rowspan="3">Erweiterung</td><td colspan="2">Die Applikation soll <Informationen> in der Darstellung gruppieren.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Informationen></td><td>• Inhalt (Text, Bilder, Bewegungen, Geräusche)
• Bedienelemente aller Art
• sonstige Bestandteile der Bedienoberfläche</td></tr>
<tr><td>Hinweis</td><td colspan="2">Es sollten möglichst zusammengehörige Informationen gruppiert werden. Als Darstellungsformen eignen sich räumliche Nähe, gleiche Farben, gleiche Formen, gleiche Schriften, gleiche Bewegungen und gleiche Klänge.</td></tr>
</table>

Nr. AI-03	**Anforderungsmuster: Ständig erreichbare Navigation**		
Metadaten	Ziel	Nutzer haben stets Zugriff auf die Navigation und damit auf die Kontrollmöglichkeiten der Applikation.	
	Grundlage	Fluent navigation on a small screen (Kaasinen 2005)	
	Abhängigkeiten	KO-02: Steuerbarkeit durch Vorliegen einer Navigation	
	Verknüpfungen		
	Konflikte		
Vorlage	Standardisierte Anforderung	Die Navigation der Applikation soll immer erreichbar sein.	
	Erweiterung	<Navigation> der Applikation soll immer erreichbar sein.	
		Parameter	Werte
		<Navigation>	• Wichtige Bedienelemente der Navigation
	Hinweis	Bei der Navigation möchten Nutzer Scrollen vermeiden. Ggf. bietet das Endgerät Zugriff auf die Navigation, ohne dass diese auf dem Bildschirm angezeigt werden muss.	

Anhang A.8 Konsistenz

<table>
<tr><th>Nr. CO-01</th><th colspan="3">Anforderungsmuster: Alle Teile einer Applikation konsistent gestalten</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer nehmen alle Bestandteile der Applikation als etwas Zusammengehöriges wahr und erleben eine flüssige, unterbrechungsfreie Benutzung.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Consistency (Preece 2000)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">CO-02: Konsistent gestaltete Funktionen,
LE-01: Lesbarer Text</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">CO-02: Konsistent gestaltete Funktionen,
CO-03: Konsistent gestaltete Informationen,
CO-04: Konsistente Darstellung auf verschiedenen Endgeräten</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Teilt sich die Applikation in verschiedene Bereiche auf, sollen die Bereiche identisch formatiert sein.</td></tr>
<tr><td rowspan="3">Erweiterung</td><td colspan="2">Teilt sich die Applikation in<Bereiche> auf, sollen <Bereiche> identisch formatiert sein.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Bereiche></td><td>• Hierarchie-Ebenen
• Gruppierungen
• Funktionen
• sonstige Teile der Applikation</td></tr>
<tr><td>Hinweis</td><td colspan="2">Wenn es nicht möglich ist, einen Bereich des Systems so zu gestalten wie die anderen Bereiche, sollte das System den Nutzer über die Zugehörigkeit dieses Bereichs informieren.</td></tr>
</table>

<table>
<tr><td>Nr. CO-02</td><td colspan="3">Anforderungsmuster: Konsistent gestaltete Funktionen</td></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Eine Funktion ist immer gleich gestaltet, egal, wo sie sich befindet. Nutzer lernen schneller die Funktionsweise und Benutzung der Applikation.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Konsistenz (Ludewig & Lichter 2010)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">WE-02: Aussagekräftige Bedienelemente, CO-01: Alle Teile einer Applikation ähnlich gestalten</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">WE-02: Aussagekräftige Bedienelemente, CO-01: Alle Teile einer Applikation ähnlich gestalten</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Bietet die Applikation an unterschiedlichen Positionen gleiche oder ähnliche Funktionen an, sollen die Funktionen überall identisch formatiert sein.</td></tr>
<tr><td rowspan="5">Erweiterung</td><td colspan="2">Bietet die Applikation an <Positionen> gleiche oder ähnliche <Funktionen> an, sollen <Funktionen> überall identisch formatiert sein.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Positionen></td><td>• Positionen innerhalb der <Navigation></td></tr>
<tr><td><Funktionen></td><td>• vom Nutzer auslösbare Funktionen
• sonstige Funktionen</td></tr>
<tr><td><Navigation></td><td>• Gesamtheit der Bedienelemente der Applikation</td></tr>
<tr><td>Hinweis</td><td colspan="2">Neben der optischen und akustischen Darstellung sollten auch die Aktionen zum Auslösen der Funktion ähnlich gestaltet sein, damit der Nutzer die Verwendung schneller lernt.</td></tr>
</table>

<table>
<tr><th>Nr. CO-03</th><th colspan="3">Anforderungsmuster: Konsistent gestaltete Informationen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Die Informationen der Applikation werden eindeutig dargestellt und der Nutzer wird nicht durch Abweichungen verwirrt.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Konsistenz (Ludewig & Lichter 2010)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">WE-02: Aussagekräftige Bedienelemente,
CO-01: Alle Teile einer Applikation ähnlich gestalten,
CO-02: Konsistent gestaltete Funktionen,
LE-01: Lesbarer Text</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">WE-02: Aussagekräftige Bedienelemente,
CO-01: Alle Teile einer Applikation ähnlich gestalten,
CO-02: Konsistent gestaltete Funktionen,
CO-04: Konsistente Darstellung auf verschiedenen Endgeräten</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Bietet die Applikation an unterschiedlichen Positionen gleiche oder ähnliche Informationen an, sollen die Informationen überall identisch formatiert sein.</td></tr>
<tr><td rowspan="5">Erweiterung</td><td colspan="2">Bietet die Applikation an unterschiedlichen <Positionen> gleiche oder ähnliche <Informationen> an, sollen <Informationen> überall identisch formatiert sein.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Positionen></td><td>• Positionen innerhalb der Navigation</td></tr>
<tr><td><Informationen></td><td>• Text
• Bilder
• Geräusche
• Bewegungen</td></tr>
<tr><td><Navigation></td><td>• Gesamtheit der Bedienelemente der Applikation</td></tr>
<tr><td>Hinweis</td><td colspan="2">Wenn Informationen in natürlicher Sprache dargestellt werden, sollte derselbe Begriff immer mit dem gleichen Wort benannt werden. Werden Informationen in Bildern dargestellt, sollte dieselbe Information immer das gleiche Bildmotiv erhalten.</td></tr>
</table>

Nr. CO-04	Anforderungsmuster: Konsistente Darstellung auf versch. Endgeräten	
Metadaten	Ziel	Die Applikation sieht auf jedem Endgerät gleich aus. Nutzer erleben unabhängig vom Endgerät die gleiche Nutzung und können die Benutzung besser lernen.
	Grundlage	Externe Konsistenz (Hix & Hartson 1993)
	Abhängigkeiten	CO-01: Alle Teile einer Applikation ähnlich gestalten, CO-02: Konsistent gestaltete Funktionen, CO-03: Konsistent gestaltete Informationen, EK-01: Beachten von Gestaltungs-Best Practices, LE-01: Lesbarer Text
	Verknüpfungen	CO-01: Alle Teile einer Applikation ähnlich gestalten, CO-02: Konsistent gestaltete Funktionen, CO-03: Konsistent gestaltete Informationen, EK-01: Beachten von Gestaltungs-Best Practices
	Konflikte	
Vorlage	Standardisierte Anforderung	Wird eine Applikation von unterschiedlichen Endgeräten aus genutzt, soll die Applikation auf jedem Endgerät identisch dargestellt werden.
	Hinweis	Selbst, wenn die zu bedienenden Endgeräte sehr unterschiedliche technische Voraussetzungen haben, lässt sich durch bestimmte Gestaltungselemente eine konsistente Gestaltung erreichen, beispielweise Farben, Formen, Formulierungen, etc.

Anhang A.9 Sichtbarkeit des Systemstatus

Nr. SI-01	**Anforderungsmuster: Systemstatus anzeigen**		
Metadaten	Ziel	Der Nutzer erkennt, was die Applikation zum aktuellen Zeitpunkt tut und welche Handlungsoptionen ihm zur Verfügung stehen.	
	Grundlage	Sichtbarkeit des Systemstatus (Nielsen 1994)	
	Abhängigkeiten	RM-01: Rückmeldung bei Aktionen, RM-02: Hinweis bei langer Bearbeitung	
	Verknüpfungen	RM-01: Rückmeldung bei Aktionen, RM-02: Hinweis bei langer Bearbeitung, RM-03: Sofortiges Anzeigen von Eingaben, KO-04: Informationen zu Vor-Einstellung	
	Konflikte		
Vorlage	Standardisierte Anforderung	Die Applikation soll den Nutzer über den aktuellen Status des aktuell ausgewählten Bereiches informieren.	
	Erweiterung	Die Applikation soll den Nutzer <Information>, <Bereich>.	
		Parameter	Werte
		<Information>	• was die Applikation macht • verfügbare Bedienelemente • verfügbare Funktionen
		<Bereiche>	• Hierarchie-Ebenen • Gruppierungen • Funktionen • sonstige Bereiche der Applikation
	Hinweis	Die Applikation sollte nur den Status der aktuell ausgewählten Hierarchie-Ebene oder Gruppierung anzeigen, um eine Informationsüberflutung zu vermeiden.	

Anhang A.10 Rückmeldung

<table>
<tr><th>Nr. RM-01</th><th colspan="3">Anforderungsmuster: Rückmeldung bei Aktionen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer erkennen unmittelbar, dass sie eine Aktion ausgeführt haben.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Feedback (Blair-Early & Zender 2008)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">SI-01: Systemstatus anzeigen,
RM-02: Hinweis bei langer Bearbeitung</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">SI-01: Systemstatus anzeigen,
RM-02: Hinweis bei langer Bearbeitung,
RM-03: Sofortiges Anzeigen von Eingaben</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Wenn der Nutzer eine Aktion durchführt, soll die Applikation ihn unmittelbar und wahrnehmbar über die ausgelöste Bearbeitung informieren.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2">Wenn der Nutzer <Aktion>, soll die Applikation ihn unmittelbar <Information>.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Aktion></td><td>• Aktivieren von Bedienelementen
• Tätigen von Eingaben</td></tr>
<tr><td><Information></td><td>• Erfolgreiches Erkennen von <Aktion></td></tr>
<tr><td>Hinweis</td><td colspan="2">Die Rückmeldung sollte so schnell wie möglich erfolgen, damit der Nutzer nicht denkt, seine Aktion sei nicht durchgeführt worden.</td></tr>
</table>

Nr. RM-02	**Anforderungsmuster: Hinweis bei langer Bearbeitung**		
Metadaten	Ziel	Nutzer erkennen unmittelbar, dass sie eine Aktion durchgeführt haben. Sie sehen ggf., wie lange die Bearbeitung der Aktion dauert.	
	Grundlage	Feedback (Blair-Early & Zender 2008)	
	Abhängigkeiten	RM-01: Rückmeldung bei Aktionen	
	Verknüpfungen	SI-01: Systemstatus anzeigen, RM-01: Rückmeldung bei Aktionen, RM-03: Sofortiges Anzeigen von Eingaben	
	Konflikte		
Vorlage	Standardisierte Anforderung	Wenn der Nutzer eine Funktion auslöst, deren Bearbeitung länger als 6 Sekunden dauert, soll die Applikation ihn unmittelbar und wahrnehmbar über die Bearbeitung informieren.	
	Erweiterung	Wenn der Nutzer <Funktion>, <Dauer>, soll die Applikation ihn unmittelbar und wahrnehmbar <Information>.	
		Parameter	Werte
		<Funktion>	• von Nutzer ausgelöste Funktion
		<Dauer>	• Anzahl Sekunden
		<Information>	• Erfolgreiches Auslösen von <Funktion> • Bearbeitungsstand von <Funktion> • Erfolgreiches Ausführen von <Funktion>
	Hinweis	Es ist im Vorhinein zu prüfen, welche Funktionen eine lange Bearbeitungsdauer haben können.	

<table>
<tr><th>Nr. RM-03</th><th colspan="3">Anforderungsmuster: Sofortiges Anzeigen von Eingaben</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer sehen unmittelbar, welche Eingaben sie in der Applikation getätigt haben.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Feedback (Scapin & Bastien 1997)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">SI-01: Systemstatus anzeigen,
RM-01: Rückmeldung bei Aktionen,
FE-02: Fehler-Benachrichtigung,
FE-05: Überprüfen von Eingaben vor Auslösen der Funktion</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">SI-01: Systemstatus anzeigen,
RM-01: Rückmeldung bei Aktionen,
RM-02: Hinweis bei langer Bearbeitung</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Wenn der Nutzer Eingaben in einem Eingabefeld macht, soll die Applikation unmittelbar anzeigen, ob und wie diese Eingaben im entsprechenden Eingabefeld erfasst wurden.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2">Wenn der Nutzer <Eingaben>, <Eingabefeld>, soll die Applikation unmittelbar anzeigen, ob und wie <Eingaben>, <Eingabefeld> erfasst wurden.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Eingaben></td><td>• Zeichen
• Werte
• Bewegungen
• Geräusche</td></tr>
<tr><td><Eingabefeld></td><td>• Eingabefeld für <Eingaben></td></tr>
<tr><td>Hinweis</td><td colspan="2">Eingaben sollten nicht angezeigt werden, wenn es sich um sensible Informationen wie z.B. Sicherheitsschlüssel handelt.</td></tr>
</table>

<table>
<tr><th>Nr. RM-04</th><th colspan="3">Anforderungsmuster: Erfolgreiche Bearbeitung anzeigen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer sehen, ob eine Funktion erfolgreich durchgeführt wurde oder nicht, nachdem sie sie ausgelöst haben.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Feedback (Scapin & Bastien 1997)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">FE-02: Fehler-Benachrichtigung,
FE-03: Aktionen rückgängig machen</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">FE-02: Fehler-Benachrichtigung,
FE-03: Aktionen rückgängig machen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Wenn der Nutzer eine Funktion ausgelöst hat, soll die Applikation ihn nach der Bearbeitung darüber informieren, ob die Funktion erfolgreich ausgeführt wurde oder nicht.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2">Wenn der Nutzer <Funktion> ausgelöst hat, soll die Applikation ihn nach der Bearbeitung <Information>.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Funktion></td><td>• von Nutzer ausgelöste Funktion</td></tr>
<tr><td><Information></td><td>• Erfolgreiches Ausführen von <Funktion>
• fehlgeschlagenes Ausführen von <Funktion></td></tr>
<tr><td>Hinweis</td><td colspan="2">Die Rückmeldung sollte aussagekräftig sein und den Nutzer darüber informieren, was in der Applikation passiert ist. So kann er die Funktionsweise der Applikation lernen.</td></tr>
</table>

Anhang A.11 Erwartungskonformität

<table>
<tr><th>Nr. EK-01</th><th colspan="3">Anforderungsmuster: Beachten von Gestaltungs-Best Practices</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer erkennen Sinn und Funktionsweise einer Funktion und können diese benutzen, ohne die Applikation zu kennen. Die Funktion ist ihnen von anderen Systemen bekannt.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Konventionen (Blair-Early & Zender 2008)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">CO-04: Konsistente Darstellung auf verschiedenen Endgeräten</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">CO-04: Konsistente Darstellung auf verschiedenen Endgeräten</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Wenn für die Gestaltung einer Funktion Best Practices existieren, soll die Applikation diese Best Practices einhalten.</td></tr>
<tr><td rowspan="5">Erweiterung</td><td colspan="2">Wenn für die <Gestaltung> von <Funktion> <Best Practices> existieren, soll die Applikation <Best Practices> einhalten.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Gestaltung></td><td>• Optik
• Akustik
• Haptik
• Aufbau/Struktur
• Ablauf
• Funktionsweise</td></tr>
<tr><td><Funktion></td><td>• von Nutzer ausgelöste Funktion
• sonstige Funktion</td></tr>
<tr><td><Best Practices></td><td>• anerkanntes, erprobtes und bekanntes Vorgehen für <Gestaltung> in anderen Systemen</td></tr>
<tr><td>Hinweis</td><td colspan="2">Eine bestehende Best Practice sollte nur verletzt werden, wenn sich dadurch ein bestimmter Vorteil ergibt oder ein Problem gelöst wird.</td></tr>
</table>

Anhang A.12 Gute Lesbarkeit

<table>
<tr><th>Nr. LE-01</th><th colspan="3">Anforderungsmuster: Lesbarer Text</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Informationen in Textform sollen unabhängig von Bildschirmgröße und Endgerät gut gelesen werden können.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Lesbarkeit (Böhringer, Bühler & Schlaich 2011)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">LE-02: Verständlicher Text,
CO-03: Konsistent gestaltete Informationen,
CO-04: Konsistente Darstellung auf verschiedenen Endgeräten</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">LE-02: Verständlicher Text,
EF-04: Verständliche Anweisungen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung 1</td><td colspan="2">Die Applikation soll für Fließtext einen Schriftgrad zwischen 9 und 12 Pixel verwenden.</td></tr>
<tr><td>Standardisierte Anforderung 2</td><td colspan="2">Die Applikation soll Schrift mit einem hohen, aber nicht absoluten Farbkontrast gegenüber dem Hintergrund darstellen.</td></tr>
<tr><td>Standardisierte Anforderung 3</td><td colspan="2">Die Applikation soll Schriften verwenden, die auf allen Betriebssystemen verfügbar sind.</td></tr>
<tr><td rowspan="3">Erweiterung 3</td><td colspan="2">Die Applikation soll eine Schriften verwenden, die auf <Betriebssystemen> verfügbar sind.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Betriebssysteme></td><td>• Betriebssysteme der Endgeräte, auf denen die Applikation funktionieren soll</td></tr>
<tr><td>Hinweis</td><td colspan="2">Bei der Lesbarkeit sind die Bildschirmgröße sowie die anzunehmende Entfernung zwischen Bildschirm und Betrachter zu berücksichtigen.</td></tr>
</table>

<table>
<tr><th>Nr. LE-02</th><th colspan="3">Anforderungsmuster: Verständlicher Text</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Nutzer verstehen die Formulierungen, die die Applikation anwendet.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Lesbarkeit (Lidwell, Holden & Butler 2003)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">LE-01: Lesbarer Text</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">EF-04: Verständliche Anweisungen,
LE-01: Lesbarer Text</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Zeigt die Applikation Informationen in Form von Text an, soll die Applikation Formulierungen verwenden, die die Nutzer verstehen.</td></tr>
<tr><td rowspan="3">Erweiterung</td><td colspan="2">Zeigt die Applikation <Informationen> in Form von Text an, soll die Applikation Formulierungen verwenden, die die Nutzer verstehen.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Informationen></td><td>• Inhalt
• Elemente der Navigation
• sonstige Information</td></tr>
<tr><td>Hinweis</td><td colspan="2">Bei der Analyse der Ziel-Nutzer sollte geprüft werden, welche Applikations-bezogenen Fachbegriffe bekannt sind und welchen Sprachgebrauch sie verwenden.</td></tr>
</table>

Anhang A.13 Hilfe anbieten

Nr. HI-01	**Anforderungsmuster: Vorliegen einer Hilfestellung**		
Metadaten	Ziel	Nutzer können bei Bedarf auf eine Hilfestellung zurückgreifen, mit der ihnen die erfolgreiche Verwendung der Applikation gelingt.	
	Grundlage	Lernförderlichkeit (Rampl 2007)	
	Abhängigkeiten	IN-03: Mehrere Aktionen gleichzeitig ausführen, EF-04: Verständliche Anweisungen	
	Verknüpfungen	EF-04: Verständliche Anweisungen, HI-02: Komplexe Funktionen mit Hilfe begleiten	
	Konflikte		
Vorlage	Standardisierte Anforderung	Die Applikation soll eine Hilfestellung anbieten.	
	Erweiterung	Die Applikation soll <Hilfestellung> anbieten.	
		Parameter	Werte
		<Hilfestellung>	• Bedienungsanleitung • FAQ • Erklärung, was Applikation macht • Erklärung, wie Applikation zu benutzen ist
	Hinweis	Je nach Art der Applikation und den genutzten Endgeräten kann die Hilfestellung in Papierform und/oder digital vorliegen.	

<table>
<tr><th>Nr. HI-02</th><th colspan="3">Anforderungsmuster: Komplexe Funktionen mit Hinweisen begleiten</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Komplexe Funktionen werden mit erklärenden Hinweisen begleitet, sodass Nutzer die erfolgreiche Verwendung der Funktion besser erlernen.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Hilfe anbieten (Scapin & Bastien 1997)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">IN-03: Mehrere Aktionen gleichzeitig ausführen,
HI-01: Vorliegen einer Hilfestellung,
KO-06: Zwei Wege der Aufgabenbewältigung,
EF-04: Verständliche Anweisungen</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">HI-01: Vorliegen einer Hilfestellung,
KO-06: Zwei Wege der Aufgabenbewältigung,
EF-04: Verständliche Anweisungen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="7">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Die Applikation soll dem Nutzer bei Verwendung einer Funktion eine begleitende Hilfestellung anbieten.</td></tr>
<tr><td rowspan="5">Erweiterung</td><td colspan="2">Die Applikation soll <Nutzer> bei Verwendung <Funktion> <Hilfestellung> anbieten.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Nutzer></td><td>• Benutzergruppe, die Benutzung von <Funktion> nicht perfekt beherrscht</td></tr>
<tr><td><Funktion></td><td>• von Nutzer ausgelöste Funktion</td></tr>
<tr><td><Hilfestellung></td><td>• Bedienungsanleitung
• FAQ
• Erklärung, was Applikation macht
• Erklärung, wie Applikation zu benutzen ist</td></tr>
<tr><td>Hinweis</td><td colspan="2">Es ist darauf zu achten, die Menge und Ausführlichkeit begleitender Hinweise an die Benutzungs-Expertise des Nutzers anzupassen: Anfänger sollen stark unterstützt werden, Experten sollen nicht verlangsamt werden.</td></tr>
</table>

<table>
<tr><th>Nr. HI-03</th><th colspan="3">Anforderungsmuster: Hilfestellung überspringen</th></tr>
<tr><td rowspan="5">Metadaten</td><td>Ziel</td><td colspan="2">Erfahrene Nutzer können die Hilfestellung für Einsteiger umgehen und erhalten damit eine schnellere, effizientere Form der Nutzung.</td></tr>
<tr><td>Grundlage</td><td colspan="2">Abkürzungen (Scapin & Bastien 1997)</td></tr>
<tr><td>Abhängigkeiten</td><td colspan="2">HI-01: Vorliegen einer Hilfestellung,
HI-02: Komplexe Funktionen mit Hilfe begleiten</td></tr>
<tr><td>Verknüpfungen</td><td colspan="2">IN-03: Mehrere Aktionen gleichzeitig ausführen</td></tr>
<tr><td>Konflikte</td><td colspan="2"></td></tr>
<tr><td rowspan="6">Vorlage</td><td>Standardisierte Anforderung</td><td colspan="2">Bietet eine Funktion Hilfestellung, sollen Nutzer die Möglichkeit haben, diese Hilfestellung zu umgehen.</td></tr>
<tr><td rowspan="4">Erweiterung</td><td colspan="2">Bietet <Funktion> <Hilfestellung>, sollen Nutzer die Möglichkeit haben, <Hilfestellung> zu umgehen.</td></tr>
<tr><td>Parameter</td><td>Werte</td></tr>
<tr><td><Funktion></td><td>• vom Nutzer ausgelöste Funktion</td></tr>
<tr><td><Hilfestellung></td><td>• begleitende Hinweise
• geführter Dialog</td></tr>
<tr><td>Hinweis</td><td colspan="2">Die Applikation kann ggf. erfassen, ob ein Nutzer sie zum wiederholten Male verwendet und die Hilfestellung automatisch ausblenden.</td></tr>
</table>